GEORGIA INSTITUTE OF TECHNOLOGY OFFICE OF CONTRACT ADMINISTRATION

PROJECT ADMINISTRATION DATA SHEET

[X] ORIGINAL [ ] REVISION NO. ____

Project No. A-3684 N.D. ELLINGSON GTRI/~~GIT~~ DATE 10/19/83

Project Director: ~~Edward J. Shanahan~~ ~~XXXXX~~/Lab ECSL/CCD

Sponsor: Army - HQ FORSCOM, Ft. McPherson, GA

Type Agreement: Delivery Order No. 1E03 under Contract F33657-82-G-2083*

Award Period: From 9/28/83 To ~~9/27/84~~ 4-30-85 (Performance) 9/27/84 (Reports)

Sponsor Amount:	This Change 5-31-85	Total to Date
Estimated:	$ 247,600	$ 247,600
Funded:	$ 247,600	$ 247,600

Cost Sharing Amount: $ ____ Cost Sharing No: ____

Title: MICROFIX Training and Application Software Development

ADMINISTRATIVE DATA OCA Contact Brian J. Lindberg X4820

1) Sponsor Technical Contact:

Mr. Charles R. France
HQ FORSCOM
Patton Hall - DCSI
Ft. McPherson, GA 30330

2) Sponsor Admin/Contractual Matters:

Mr. Staten Corbett ONR
Contracting Division
Building 184
Ft. McPherson, GA 30330
752-2469

Defense Priority Rating: DO S-1 Military Security Classification: N/A

(or) Company/Industrial Proprietary: N/A

RESTRICTIONS

See Attached Government Supplemental Information Sheet for Additional Requirements.

Travel: Foreign travel must have prior approval – Contact OCA in each case. Domestic travel requires sponsor approval where total will exceed greater of $500 or 125% of approved proposal budget category.

Equipment: Title vests with Gov't except that items costing $1,000 or less vest with Research providing prior written approval to purchase received from Contracting Officer.

COMMENTS:

* Basic Ordering Agreement No. F33657-82-G-2083 on file with Lab Director's Office/ECSL.

Note: Funds listed for each task in Delivery Order is maximum expenditure for that task.

COPIES TO:

Project Director (Shanahan)
Research Administrative Network
Research Property Management
Accounting
~~Procurement~~/EES Supply Service
~~Security~~ Services
Reports Coordinator (OCA)
Research Communications (2)
GTRI ~~Library~~
Project File
Other I. Newton

GEORGIA INSTITUTE OF TECHNOLOGY — OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION/CLOSEOUT SHEET

Date October 2, 1985

Project No. A-3684 — School/Lab ECSL/CCD

Includes Subproject No.(s) N/A

Project Director(s) N.D. Ellingson — GTRC

Sponsor Army HQ, Forscom; Ft. McPherson, Ga.

Title MICROFIX Training and Application Software Development

Effective Completion Date: 5/31/85 (Performance) _____ (Reports)

Grant/Contract Closeout Actions Remaining:

- [ ] None
- [X] Final Invoice or Final Fiscal Report
- [X] Closing Documents
- [X] Final Report of Inventions – Patent Questionnaire Sent to Project Director
- [X] Govt. Property Inventory & Related Certificate
- [ ] Classified Material Certificate
- [ ] Other _____

ontinues Project No. _____ Continued by Project No. _____

OPIES TO:

oject Director
esearch Administrative Network
esearch Property Management
ccounting
ocurement/GTRI Supply Services
esearch Security Services
ports Coordinator (OCA)
gal Services

Library
GTRC
Research Communications (2)
Project File
Other Heyser

Jones

RM OCA 69.285

A-3684

PLAN OF INSTRUCTION FOR MICROFIX TRAINING COURSE FOR TOPOGRAPHIC PERSONNEL FT. BELVOIR, VIRGINIA

December 12 - 16, 1983

Prepared by

Engineering Experiment Station
Georgia Institute of Technology

for

Assistant Chief of Staff, Intelligence
U.S. Army Forces Command
Ft. McPherson, Georgia

and

Belvoir R&D Center
Ft. Belvoir, Virginia

November 30, 1983

PLAN OF INSTRUCTION FOR MICROFIX TRAINING COURSE FOR TOPOGRAPHIC PERSONNEL

1.0 PURPOSE

This Plan of Instruction is for a one-week short course covering the MICROFIX System I hardware and software capabilities. All aspects of the system will be presented, nevertheless, the major emphasis will be on those capabilities designed to assist the terrain analyst. The goal is twofold: first, to provide the users with the understanding and ability to utilize the system; and second, to elicit feedback from participants to aid in future system modification and development.

2.0 BACKGROUND

MICROFIX is a Quick Reaction Capability (QRC), established by the QRC #55 Management Action Plan, dated January 26, 1983, which is being developed by the Army Intelligence Community to automate tactical intelligence related functions prior to the fielding of ASAS. User validated information derived from MICROFIX will be fed to the Joint Tactical Fusion Program (JTFP) to assist in the development of ASAS.

2.1 Design Philosophy

The overall design philosophy is designed to force a closer interaction between the operational and technical aspects of system development. This approach was established after lengthy examination of the traditional top-down system development method which assumes complete understanding of all requirements at the initiation of the project. In reality, requirement statements are excellent guidelines for system development but are not in sufficient detail to support coding of software. Needed was a bottom-up evolutionary approach which employs a technique to establish a mature capability resulting from constant user/developer interaction. The MICROFIX program uses the "use-learn-develop" approach by establishing a software capability in its infant form; by obtaining feedback from analysts from operational field units who have used the system; and by rewriting the software to meet the expressed needs of the analysts. This cycle is repeated until the mature capability is established. Two groups learn by using this technique: the software developer has a more precise definition of the requirement and the analyst is educated in the capabilities and limitations of computer systems. The key is to maintain the dialogue between those two groups without the administrative delays caused by system change requests and other formal documentation employed in the traditional system development methodology. In a sense, this course is a part of this methodology, and feedback should be expected.

2.2 Hardware Configuration

MICROFIX System I consists of the following commercial data processing items and is manufactured to meet the NACSIM 5100A criteria of TEMPEST: a Central Processing Unit (CPU) with a minimum of 128K RAM, a full 128 ASCII set keyboard, joystick, dual 5-1/4" floppy disk drives, monochrome monitor, color monitor, dot matrix graphics printer, graphics entry device, laser video disk player, Winchester hard disk (20 MB), and a power conditioning system (110 and 220 models). Optional components consist of a paper tape reader/punch (PTR/P) and video cassette recorder (VCR). The CPU is the core of the system. It contains the Main Logic Board Assembly equipped with the various logic printed circuit boards and the interface cards for host and peripheral devices. The CPU performs the circuitry logic inherent to the microprocessor system. Connecting to the CPU are the plug-in diskette drives, keyboard, and monitors. The drivers are contained in the CPU assembly and can be easily removed and replaced. They accept the floppy diskettes which are used in processing data. There are two monitors in the system, one to display text and one for displaying graphics information. The keyboard plugs into the front panel of the CPU. Its purpose is to allow the operator to key in information and control joystick operations relative to the graphics and may displays. The keyboard is equipped with an accessible enhancer board that provides special operational functions such as auto repeat keying and lower case letters. The hard disk drive is for mass random access storage. When provided with a Mirror Board, it is able to transfer to and receive data from a VCR. The VCR is an add-on peripheral device that provides backup mass storage for the system. The graphics tablet is an output device to control and mark graphic representations. The power conditioner provides power overvoltage and power spike protection. There will be two models: one type for standard 60 Hz, 110 VAC, and the other for 50 Hz, 220 VAC. The video disk player stores and displays basic tactical maps. The printer provides a hard copy paper printout. The printer may, depending on operational requirements, be substituted for by a PTR/P. The PTR/P transmits and receives digital data.

2.3 Software

MICROFIX System I software is primarily a database management system for storage, manipulation, and display in text and graphic format. It is written in two languages, PASCAL and dBASE II, a database management language. The six major components include:

1. Information Retrieval System (IRS) -- enters, sorts, and maintains data in the database and generates usable battlefield reports.

2. Graphical Intelligence Analysis System (GIAS) -- allows the operator to spatially display order-of-battle and spot reports, roam across the video disk based maps, input sketches, and declutter the screen as required.

3. Collection Planning Aid (CPA) -- allows generation and maintenance of the collection plan, and formats tasking requests. Subsequent features include system selection, map display of collection assets, and coverage and collection reports.

4. Topographical Support (TOPO) -- provides for the building of a document related database and the retrieval of information by geograpic area, subject, and other ancillary parameters. Additional software is currently being developed to evaluate battlefield environmental effects and for a slope analysis.

5. Commercial word processing and formatting packages.

6. System diagnostic test programs.

3.0 PLAN OF OPERATION

3.1 Instructional Personnel

The instructional team leader is Mr. Michael Rowan, a Research Scientist at Georgia Tech. Mr.Rowan has a background in geographic databases and spatial analysis. He is responsible for the design of the topographic library subsystem, and he is evaluating the feasibility of incorporating other terrain related capabilities into the MICROFIX system. Other members include: Mr. Ben Atha, a Research Scientist, who has worked on both hardware and software aspects of MICROFIX, including videodisk map graphics software; and Mr. John Peck, a Programmer, who has worked on the topographic library routines and the slope analysis software.

3.2 Location of Instruction

The course will be taught at the Defense Mapping School, Ft. Belvoir, Virginia from December 12 through December 16, 1983. The Government will furnish four MICROFIX systems and one VCR for the course.

3.3 Instructional Methodology

The course will cover both hardware and software elements of the system. The training will parallel procedures and topics developed for the New Equipment Training (NET) team courses being taught at other installations with two major exceptions. First, the NET course covers a two-week period, whereas the Ft. Belvoir course will be accomplished in one week. It will be necessary to reduce the amount of detail for some areas. Second, the NET course is primarily oriented toward the military intelligence analyst. In contract, the Ft. Belvoir course will focus more on applications related to the terrain analyst. Terrain related software, not yet implemented on the operational system, will also be demonstrated at this session.

Classroom procedures will consist of both lectures and practical exercise by the participants, with the emphasis on the latter. The small number of participants (approximately 12) is conducive to a high degree of student/instructor interaction and for a sufficient amount of hands-on experience with the computer. It is expected that these conditions will lead to feedback from the participants concerning ease of usage and overall capabilities of the topographic related software.

3.4 Typical Teaching Day

0800 - 0850	Period One
0900 - 0950	Period Two
1010 - 1100	Period Three
1110 - 1200	Period Four
1200 - 1300	Lunch
1300 - 1350	Period Five
1400 - 1450	Period Six
1510 - 1600	Period Seven
1610 - 1700	Period Eight

4.0 OUTLINE OF TOPOGRAPHIC PERSONNEL COURSE

1st Day

1 hour I. INTRODUCTION

A. Overview of MICROFIX

1. Background
2. Outline of Course
3. Introduction to MICROFIX System I Hardware

B. Absolute Warnings

1. Handling of Boards, Cabling, and Connectors
2. Operations in a Hostile Environment

3 hours II. SYSTEM INITIALIZATION AND DIAGNOSTICS

A. Power Considerations

B. Startup Procedures

C. Diagnostic Procedures

D. System Check

4 hours III. CORVUS OPERATIONS AND BACKUP

A. Overview

1. Lights
2. Front Panel Switches
3. Boards/Equipment
4. Cabling
5. Physical Organization
6. Volume Organization

B. Initialization

1. User Relationship to Volume/OS
2. Mirror
3. Backup and Loading Software from Backup

2nd Day

1 hour IV. INTRODUCTION TO COMPUTERS

A. Software

B. Operating Systems

C. Apple Computer

D. Database Management Systems

1 hour V. MICROFIX OVERVIEW

A. IRS

B. GIAS

C. CPA

D. TOPO

6 hours VI. TOPOGRAPHIC LIBRARY ROUTINE

A. Background

B. Record Structure and Indexing

C. Coding Standards

D. Data Input and Editing

E. Area Search

3rd Day

F. Listing

G. Update

8 hours VII. OTHER TERRAIN RELATED ACTIVITIES

A. Slope Analysis

1. Data
2. Processing Techniques
3. Display

B. Battlefield Environmental Effects System

1. Overview
2. SWO Comments
3. Demonstration

C. General Discussions

4th Day

3 hours VIII. IRS (Information Retrieval System)

A. Spot Reports

B. Order of Battle

C. TOE/Attrition

D. Initialization

E. Backup

F. Command Line

G. Coordinate Conversion

3 hours IX. GIAS (Graphic Intelligence Analysis System)

A. Area Selection

B. Declutter

C. Roam

D. Reports

E. Location History

F. Create Sketch

G. Display Sketch

1 hour X. SOFTWARE CUSTOMIZATION

A. Symbols

B. Text Edit

C. Default Values

1 hour XI. APPLICABILITY OF OTHER FUNCTIONS TO TOPO PROBLEMS

5th Day

3 hours XII. CPA (Collection Planning Aid)

A. Collection Plan

B. Tasking/Requests

C. System Selection

D. Coverge Map

E. Collection Reports

3 hours XIII. OTHER SOFTWARE PACKAGES

A. Format Generator

B. WordStar

2 hours XIV. REVIEW AND DISCUSSIONS

5.0 MATERIALS

The training team will provide copies of all visual aids and lesson plans to the participants during the course.

MICROFIX SYSTEM ONE

RELEASE DOCUMENTATION

(BEES)

Battlefield Environmental

Effects System

Prepared by

Command and Control Division
Electronics and Computer Systems Laboratory
and the
Electro-Optics Division
Electromagnetics Laboratory
Engineering Experiment Station
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia 30332

September, 1984

TABLE OF CONTENTS

Section 1

OVERVIEW

BATTLEFIELD ENVIRONMENTAL EFFECTS SYSTEM (BEES)

OVERVIEW

This software was adapted from programs in the Battlefield Environmental Effects System (BEES) which the Army Engineer Topographic Laboratories developed and implemented in BASIC on a Hewlett-Packard HP-85 microcomputer.

This system consists of a set of individual MT+ PASCAL programs connected by chaining. A driver program displays the main system menu on the user's terminal and either chains in an executable application program or exits to the calling level depending on the menu option chosen by the user. The programs associated with driver menu options contain chain statements to effect a return to the driver when they are terminated; the main menu is then redisplayed.

An individual program comprises a number of relocatable modules which are linked together to form a binary object (executable) program. Each linkable module is set up by putting the source code for a group of procedures and functions in a single disk file and compiling them as a unit. In this system procedures/functions are lumped together primarily based on logical relationship (e.g., to form a library or perform input functions for a given program). Modularization accommodates limitations on the amount of code the PASCAL compiler can compile at one time or limitations on the amount of text the SPP text editor can handle in a single source file. An overlay structure may be required if the amount of relocatable (compiled) code which must be linked together exceeds the upper limit on addressable memory space, 64K bytes. For this application the use of overlays has not so far been necessary.

FILENAME EXTENSIONS

Filenames under CP/M have the form NAME.EXT, where NAME usually serves as an application identifier and EXT describes the file's "type"; e.g., source file, relocatable, etc. The extensions used for BEES are listed below. They were chosen to conform with standard use in other MICROFIX software or enforced by system software requirements.

.BAK	backup source files created by SPP text editor
.BLD	PASCAL librarian command files for LIBMT
.CMD	PASCAL linker command files
.COM	executable object files produced by LINKMT
.ERL	relocatable object files produced by PASCAL compiler
.MNU	menu text files
.SRC	PASCAL MT+ source files

MT+ SYSTEM LIBRARIES

The following relocatable libraries must be linked with other relocatable modules to create executable programs for the system. Some may not be needed by some programs. Necessary libraries are specified in the individual program descriptions.

FPREALS.ERL	floating point numbers
PASLIB.ERL	basic PASCAL library (comparisons, I/O, etc.)
RANDOMIO.ERL	random disk file I/O
TRANCEND.ERL	PASCAL transcendental functions

MICROFIX SYSTEM LIBRARIES

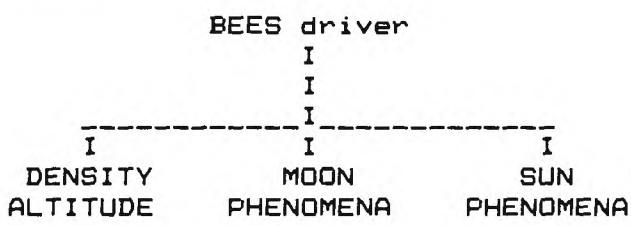
MICROFIX library CURLIB.ERL contains utility procedures for handling I/O for the terminal screen. The following procedures from CURLIB are used by BEES and declared EXTERNAL in modules which call them:

CLREOS	clear to the end of the terminal screen
CLRSCR	clears the terminal screen
CLRLN	clears the current line on the screen
GOTOXY(X,Y: BYTE)	moves cursor to position X,Y on the terminal screen; 0,0 is upper left corner.
MENU(NAME: STRING)	reads menu text from disk file NAME and displays the text on the screen

Section 2

ORGANIZATION

BEES SYSTEM STRUCTURE



BEES SYSTEM LIBRARY

Relocatable library BEESLIB.ERL contains BEES procedures and functions used by more than one application program.

Source files:

IOMOD.SRC	I/O-related procedures or functions
MATHMOD.SRC	mathematical procedures or functions

Functions (MATHMOD.SRC)

The following REAL functions were implemented because PASCAL has no intrinsic arcsin and arccos functions and no mod function with real arguments, and the BASIC programs from which this software was adapted used degrees as trig function arguments and outputs whereas PASCAL uses radians. Source code for these functions is in file MATHMOD.SRC.

Function	Purpose
ASIN(ARG: REAL)	ARCSIN(ARG), radian output
ACOS(ARG: REAL)	ARCCOS(ARG), radian output
DTR(ARG: REAL)	convert degrees to radians
RMOD(ARG,MODF: REAL)	modulus with real arguments
RTD(ARG:REAL)	convert radians to degrees

Procedures (MATHMOD.SRC)

GETLOCALTIME(LONG:REAL) Calculate time difference from LONG to Greenwich meridian and display on screen.

Procedures (IOMOD.SRC)

GET_NEXT_KEY_PRESSED(VAR C: CHAR)
Get value for next keyboard character entry

GET_A_YESNO_ANSWER(VAR C: CHAR)
As above but only accept and echo Y or N (lower case input is changed to upper case)

GET_MENU_CHOICE(MX:INTEGER;VAR C:INTEGER)
Read keyboard input (up to 2 characters) followed by RETURN. Accept only integers between 1 and MX, or RETURN by itself which defaults to 0. Return result in C.

PAUSE

Produce a screen prompt for keyboard input before displaying more output. Used to pause between screen changes.

Functions (IOMOD.SRC)

CHAR_TO_INTEGER(STR:STRING): INTEGER

Convert a string to integer equivalent.

CHAR_TO_REAL(LINE:STRING): REAL

Convert a string to a real number.

INTIN(PROMPT:STRING;MIN,MAX:INTEGER): INTEGER

Use PROMPT to solicit integer input in interval [MIN,MAX].

KEYPRESSED: BOOLEAN

Detect keyboard input.

REALIN(PROMPT:STRING;MIN,MAX:REAL): REL

Use PROMPT to solicit real input in interval (MIN,MAX).

Library creation

Generation of linkable relocatable library BEESLIB.ERL requires compiling source code for individual modules to be included in the library and then processing the relocatable modules with the MT+ library processor LIBMT. LIBMT expects input in the form of a text file which contains the names of the output library file and all the relocatable modules to be entered in the library. The name of this text file must have extension BLD, as explained in the MT+ PASCAL manual. The linker LINKMT can be instructed to include only library modules containing routines which are called by the program the library is being linked with in the executable code it generates. The BEES library currently has two modules, one for mathematical routines and one for I/O related routines.

To generate library BEESLIB.ERL:

1. Compile source files BEESMOD.SRC and MATHMOD.SRC
2. Prepare (if necessary) text file BEESLIB.BLD containing
   BEESLIB.ERL
   IOMOD.ERL
   MATHMOD.ERL
3. Use the command: LIBMT BEESLIB

BEES SYSTEM DRIVER

Source file:

BEES.SRC	Driver to display main menu and chain to other system programs.

Procedures: main procedure

Data files:

BEES.MNU	Contains text of main menu. This is an ASCII file created with a text editor.

Relocatable modules needed to link BEES:

BEES.ERL, BEESLIB.ERL, CURLIB.ERL, PASLIB.ERL

Linker command file: LKBEES.CMD

Adding a new option requires:

1. Adding the option to the menu text in file BEES.MNU.
2. Adding an appropriate chain statement to the source code in BEES.SRC. Also the MX paramater in the call to GETMENUCHOICE must equal the largest menu option number on the main menu.
3. Compiling BEES.SRC and linking the BEES program.
4. Adding a 'chain back' statement to the program to be used as the new option, compiling and linking it.

BEES SYSTEM OPTIONS

DENSITY ALTITUDE

Discussion:

This program computes density altitude from user supplied input data and can also compute loading capacity for certain helicopter models at the current density altitude. The program was adapted fairly straightforwardly from the analogous ETL BASIC program. and all possible comments and variable names have been preserved or clarified in the PASCAL version. Theoretical discussion of the equations used by the program is beyond the scope of this document. Parameters for the implemented helicopter models were obtained from the ETL BASIC program.

Source files:

COPTER.SRC	Copter menu call; calculations for some helicopters.
COP2.SRC	Calculations for remaining helicopters.
DENSALT.SRC	Input and density altitude computations.

Procedure	Source file
AH_IG	COPTER.SRC
AH_1S	COPTER.SRC
CALCULATE_DENSITY_ALTITUDE	DENSALT.SRC
CH_47A	COPTER.SRC
CH_47B	COPTER.SRC
CH_47C	COPTER.SRC
CH_54A	COP2.SRC
CH_54B	COP2.SRC
COMPUTE_A_LOAD_CAPACITY	COPTER.SRC
DO_DENSITY_ALTITUDE_CALCULATIONS	DENSALT.SRC
DO_LOAD_CAPACITY_CALCULATIONS	COPTER.SRC
Main	DENSALT.SRC
OH_6A	COP2.SRC
OH_58A	COP2.SRC
OH_58C	COP2.SRC
GET_DENSITY_ALTITUDE_INPUTS	DENSALT.SRC
GET_PALTITUDE_FROM_PRESSURE	DENSALT.SRC
GET_PRESSURE_FROM_PALTITUDE	DENSALT.SRC
HEADING	DENSALT.SRC
PRINT_DENSITY_ALTITUDE	DENSALT.SRC
PRINT_OUTPUT	COPTER.SRC
SETUP_LOAD_CAPACITY_CALCULATIONS	COPTER.SRC
UH_1CM	COP2.SRC
UH_1DH	COP2.SRC
UH_60A	COP2.SRC

Relocatable modules needed to link DENSALT:

DENSALT.ERL, COPTER.ERL, COP2.ERL, BEESLIB.ERL,
CURLIB.ERL, FPREALS.ERL, TRANCEND.ERL, PASLIB.ERL

Linker command file: LKDENS.CMD

Adding a new helicopter requires:

1. Adding the model to the helicopter menu in HELI.MNU.
2. Adding a procedure with appropriate calculations to source file COP2.SRC.
3. Adding the new case to the CASE statement, procedure COMPUTE_A_LOAD_CAPACITY in file COPTER.SRC.
4. Compiling COPTER.SRC and COP2.SRC and linking DENSALT.

Notes:

Source file COPTER.SRC got too big for the SPP editor so the rest of the helicopter calculations were placed in file COP2.SRC.

Density Altitude is close to running out of memory when it is linked. Adding more than a few lines of code may require space reduction elsewhere in the program.

MOON PHENOMENA

Discussion:

The Moon Phenomena program has two distinct functions: calculate moonrise/moonset times for a specific date and geographic location between 60 deg N and 60 deg S for the current year, or output dates for moon phases for a specific month in the current year. The operator inputs date/location information but the program must have access to additional data about the moon's movements which must be updated annually. The source for the latter data is the Almanac for Computers available from the Naval Almanac Office of the U.S. Naval Observatory in Washington, D.C. This document contains the necessary data and information about the algorithm used to calculate moonrise/moonset times.

Monthly phase dates can be acquired by the program by straightforward retrieval. The dates are not calculated, but extracted from the almanac and put somewhere the program can find them. Since each of 12 months has 4 or 5 dates when moon phases (full, new, waxing crescent, waning crescent) occur, this means storage of about 50 data items. The most obvious way to do this in FORTRAN or BASIC would be to store them in a 1 or 2 dimensional array by using DATA statements and then get the appropriate numbers out of the array when needed. Alas, PASCAL has nothing like a DATA statement and keeping these numbers in the program itself would require 50 separate assignment statements.

The moonrise/set times are arrived at via an iterative process which employs (as of the 1984 Almanac) a 7th order series expansion. The data set for the algorithm used in the 1984 almanac consists of 768 separate floating point coefficients which may be used in the series expansion (16 each for 48 "weeks") and several constants which apply to the year itself.

The "first draft" PASCAL version of this program avoided the need for separate assignment statements by using a sequential access disk file to store all the almanac data. The use of a data file reduces the size of the program and has the additional attractive aspect of eliminating the need to edit, recompile and relink the program when the data must be modified. The data file can be updated by using a text editor. However, use of a sequential file causes the program to execute rather slowly since it has to read every number in the file until it gets to the ones it needs. The slowness problem was eliminated by using a separate program to produce a random access file which contains essentially the same data in a somewhat different format.

Program RANREAD

This program reads a sequential disk file containing a year's worth of moon data and writes a random access disk file containing the same data in a modified format. Input file name is MOONDAT, output filename is MOONRAN. MOONDAT is created by extracting appropriate data from the Almanac for Computers and putting it in the file with a text editor. Specifications for both data files are included here. RANREAD needs to be executed only to establish the MOONRAN file.

Source file: RANREAD.SRC

Procedures:

Main procedure
MOVE_DATA_TO_RANDOM_FILE

Data files:

MOONDAT- Contains moon data for current year, extracted from the Almanac for Computers. This is an ASCII text file which must be created by using a text editor. The RANREAD program reads it and writes the data to random access file MOONRAN. The Moon Phenomena program gets data from MOONRAN. The file structure for MOONDAT is as follows:

line 1: 4 digit year (e.g. 1984)
line 2: "A" value for the current year (4 in 1984)
line 3: "D5" value for the current year (347.81 in 1984)
line 4: month name (first is JANUARY)
line 5: 6 integers separated by spaces.

Integer 1 is a code number the program will use to decide what column headings to output for phase date info for this month. Some months have 4 phase dates, others have 5. The dates are most reasonably output in increasing order. The codes are as follows:

1	WANING, NEW, WAXING, FULL
2	NEW, WAXING, FULL, WANING
3	WAXING, FULL, WANING, NEW
4	FULL, WANING, NEW, WAXING
10	WANING, NEW, WAXING, FULL. WANING
20	NEW, WAXING, FULL, WANING, NEW
30	WAXING, FULL, WANING, NEW, WAXING
40	FULL, WANING, NEW, WAXING, FULL

Integers 2 - 6 are days of the current month when the phases represented by the codes above occur. Integer 6 should be zero for months with only 4 phase dates.

line 6: 8 coefficients for moon GHA power series for week 1 of this month, in order c7....c0 and separated by spaces.

line 7: 8 coefficients for moon declination power series for week 1 of this month, in order c7...c0 and separated by spaces.

line 8: same as line 6, for week 2.
line 9: same as line 7, for week 2.
line 10: same as line 6, for week 3.
line 11: same as line 7, for week 3.
line 12: same as line 6, for week 4.
line 13: same as line 7, for week 4.

. Repeat lines 4 - 13, 11 more times
. for the 11 remaining months.

****

IMPORTANT NOTES

1. For formatting the MOONDAT file, numbers on the same line must be separated by SPACES. Commas are NOT valid separators.

2. All decimal points must be embedded between 2 visible digits. Leading and trailing decimal points are unacceptable to the PASCAL compiler. (This applies to numbers within programs or numbers to be read using standard PASCAL I/O.)

Examples:	ACCEPTABLE	NOT ACCEPTABLE
	0.5	.5
	332.0	332.
	-0.7	-.7
	0.13E-5	.13E-5

****

A copy of the 1984 MOONDAT data file appears at the end of this section.

MOONRAN - random access moon data file produced by RANREAD.

This file's structure is defined by its record definition in programs RANREAD and MOON. All records in the file have the same structure and exactly the same length, and are stored contiguously on the disk. Instead of reading sequentially through the file to locate data, the random I/O handler calculates the

physical location of a record by its record number and starts reading data at that location. I/O is fast and any record can be found in the same length of time. Random files and random file I/O are discussed in the MT+ PASCAL manual on pp. 184 - 188. The record structure for the 12-record MOONRAN file is as follows:

```
    MONTHNAME: STRING[9];
    YEAR:      INTEGER;
    RA:        INTEGER; ("A" constant for current year)
    RD5:       REAL;    ("D5" constant for current year)
    WD:        INTEGER; (1st integer from 'line 5' of MOONDAT)
    PHASEDATE: ARRAY[1..5] OF INTEGER; (rest of 'line 5')
    MONTHDATA: ARRAY[1..4,1..16] OF INTEGER;  (coefficients)
```

Refer to the MT+ PASCAL manual and program listings of RANREAD.SRC and MOON.SRC to see how this file is used.

Relocatable modules needed to link RANREAD:

RANREAD.ERL,FPREALS.ERL,RANDOMIO.ERL,PASLIB.ERL

Linker command file: none

Program MOON

Source files:

INMOON.SRC	Performs date/location input.
MOON.SRC	Main/computation module for moon phenomena.

Procedure	Source file
ASSIGN_MONTHS	INMOON.SRC
CHOOSE_PROGRAM_OPTION	MOON.SRC
FIND_DATA	MOON.SRC
GET_DATE_AND_LOCATION	INMOON.SRC
GET_MONTH_AND_PRINT_PHASES	MOON.SRC
GHA_AND_DEC	MOON.SRC
ITERATE	MOON.SRC
Main procedure	MOON.SRC
PHENOMENA_TIME_CALCULATIONS	MOON.SRC
PRINT_MOON_HEADING	MOON.SRC
PRINT_OUTPUT	MOON.SRC
SETUP	MOON.SRC

Data files:

MOON.MNU	Contains text of moon phenomena option menu.
MOONRAN.SRC	Moon data input for current year. This file was discussed along with the RANREAD program.

Relocatable modules needed to link MOON:

MOON.ERL, INMOON.ERL, BEESLIB.ERL, CURLIB.ERL, FPREALS.ERL, TRANCEND.ERL, RANDOMIO.ERL, PASLIB.ERL

Linker command file: LKMOON.CMD

SUN PHENOMENA

Discussion:

This program computes sun phenomena (sunrise/sunset, civil twilight, nautical twilight, and astronomical twilight) for user specified date and geographic location between 60 deg N and 60 deg S. It requires no additional data or annual update. The algorithms used are described in the Almanac for Computers.

Source files:

INSUN.SRC	Input module for sun phenomena.
SUNRISE.SRC	Main processing module for sun phenomena.

Procedure	Source file
ASSIGN_MONTHS	INSUN.SRC
COMPUTE_PHENOMENA_TIMES	SUNRISE.SRC
FIND_RIGHT_ASCENSION_AND_DECLINATION	SUNRISE.SRC
GET_DATE_AND_LOCATION	SUNRISE.SRC
GET_INPUTS	INSUN.SRC
Main procedure	SUNRISE.SRC
PRINT-SUN-HEADING	SUNRISE.SRC

Relocatable modules needed to link SUNRISE:

SUNRISE.ERL, INSUN.ERL, BEESLIB.ERL, CURLIB.ERL, FPREALS.ERL, TRANCEND.ERL, PASLIB.ERL

Linker command file: LKSUN.CMD

Section 3

LISTINGS

Data File

```
 1: 1984
 2: 4
 3: 347.81
 4: JANUARY
 5: 2 3 11 18 25 0
 6: 1956.6967 1392.2494 4.7149 1.0619 -1.3222 -0.041 0.1996 -0.0365
 7: -23.0755 8.7563 8.1968 -1.9327 -0.3374 0.3147 -0.0184 -0.0396
 8: 1512.9702 1395.8771 -6.3791 -2.7118 0.4453 0.5095 0.1491 -0.0126
 9: 12.013 19.3377 -2.9381 -2.6935 -0.7391 -0.1018 0.1222 0.0513
10: 1401.7515 1386.0607 7.0169 -0.8042 -2.1805 0.9104 0.2125 -0.213
11: 14.7271 -21.1826 -6.7235 4.9178 -0.0434 -0.7043 0.1862 0.0636
12: 1664.3651 1389.577 -1.1272 2.0618 0.8783 -0.4557 -0.1409 0.0379
13: -24.0859 -6.7991 9.8414 0.6801 -0.7801 -0.1219 0.0444 0.0213
14: FEBRUARY
15: 2 1 10 17 23 0
16: 1583.2818 1400.0894 2.3508 -1.9301 -0.3858 0.102 -0.0344 0.0177
17: -9.3321 18.9236 2.7345 -1.7525 0.142 -0.0355 -0.0326 0.0029
18: 1494.3531 1382.9185 -6.9839 3.3789 2.4793 -0.4741 -0.5385 -0.0039
19: 24.7242 5.5618 -11.9805 -3.141 1.1162 0.8981 -0.029 -0.1379
20: 1743.5885 1391.4612 0.61 -2.2693 0.5491 0.3466 -0.08 -0.0014
21: -7.1489 -23.6776 4.4485 3.1948 -0.7645 0.0781 0.0332 -0.0383
22: 1644.8979 1393.9355 4.4829 0.3412 -0.9961 0.059 0.1217 -0.034
23: -21.9627 10.8349 7.4573 -1.0591 -0.2096 0.1786 0.0223 -0.0171
24: MARCH
25: 2 2 10 17 24 0
26: 1593.0264 1400.9025 -0.428 -2.1974 -0.1539 0.0678 0.0026 0.0113
27: -0.5209 20.5402 0.4576 -1.7649 -0.048 -0.0877 -0.0124 0.0072
28: 1497.7162 1381.5885 -1.2897 4.6442 -0.0689 -1.3327 0.0084 0.234
29: 25.0777 -5.121 -13.3505 -0.2399 1.9232 0.2531 -0.2229 -0.036
30: 1750.8485 1389.1137 -1.572 -0.4229 1.1711 0.1417 -0.1366 -0.0216
31: -16.0886 -19.4531 8.3071 2.5415 -0.9222 -0.0616 0.0189 0.0049
32: 1652.7643 1397.8724 3.8831 -1.364 -0.5177 0.2235 -0.108 -0.0292
33: -16.0829 16.3826 4.9265 -1.8491 0.0788 -0.019 -0.0383 0.0227
34: APRIL
35: 2 1 9 15 23 0
36: 1580.8114 1394.0368 -6.1669 -1.2152 0.9618 -0.4734 -0.0222 -0.0827
37: 17.8407 16.3132 -5.7978 -2.7625 -0.2204 0.1394 0.0891 0.0101
38: 1474.0227 1388.4301 3.1077 -1.529 -1.0297 0.5295 0.0683 -0.0578
39: 10.9676 -23.7277 -5.6196 4.2293 0.471 -0.2849 0.0133 0.0041
40: 1730.4134 1387.148 3.0262 2.8268 -0.6343 -0.6551 0.1302 0.0737
41: -25.8676 -0.8498 11.0932 -1.1508 -0.9491 0.3261 0.0612 -0.0407
42: 1642.1874 1400.4176 -1.6860 -2.2857 0.0006 0.0481 0.047 0.0397
43: 1.9147 21.0725 0.0251 -1.903 -0.2588 -0.1089 0.0219 0.02
44: MAY
45: 2 1 8 15 22 0
46: 1572.6964 1385.2009 -3.963 3.4604 1.2123 -0.7677 -0.27 0.0921
47: 25.5536 4.325 -11.5661 -1.7789 1.2048 0.3956 -0.1147 -0.0641
48: 1468.1854 1391.1379 -1.8198 -2.2545 0.6303 0.4641 -0.0107 -0.0245
49: -5.6221 -24.9319 2.575 4.014 -0.0204 -0.1469 -0.0657 -0.0235
50: 1724.1429 1392.9074 6.284 -0.3137 -1.3945 0.3316 0.1312 -0.0704
51: -22.0919 11.7944 7.4617 -2.4286 0.1157 0.2405 -0.1031 -0.0026
52: 1641.1095 1394.8413 -6.4668 -1.712 0.957 0.5667 0.0309 -0.0839
53: 15.4139 18.3217 -4.6046 -2.9777 -0.4553 0.1163 0.1227 0.0297
54: JUNE
55: 3 6 13 21 29 0
56: 1547.4293 1387.2098 5.3086 0.0616 -2.0493 0.4293 0.2691 -0.1264
57: 18.9816 -17.4697 -8.8187 3.3523 0.4594 -0.4374 0.0864 0.0448
58: 1448.3706 1386.1382 -2.6357 2.7725 1.5622 -0.5559 -0.2966 0.043
59: -23.266 -11.135 11.0061 1.7642 -1.2924 -0.323 0.12 0.0635
60: 1712.2374 1400.7844 1.7076 -2.5578 -0.1633 0.0884 -0.045 0.0233
```

```
 61: -6.6623 20.132 1.9087 -1.5841 0.1308 -0.1268 -0.0221 -0.003
 62: 1620.9773 1382.3238 -3.5903 5.1006 1.247 -1.4171 -0.3094 0.2436
 63: 25.8664 1.9682 -12.9292 -1.6094 1.8573 0.4958 -0.2486 -0.0818
 64: JULY
 65: 3 5 13 21 28 0
 66: 1539.9118 1392.5226 2.3683 -2.9066 -0.2307 0.4225 -0.0713 0.03
 67: 3.6871 -24.7146 -1.0377 3.5689 -0.3119 0.0299 0.0682 -0.03
 68: 1439.2353 1387.7747 4.0017 3.0247 -1.2247 -0.6589 0.2566 0.081
 69: -25.5211 4.2716 10.5729 -1.7028 -0.9885 0.4038 0.0924 -0.0686
 70: 1713.1439 1400.0512 -3.2676 -2.745 0.0114 0.1602 0.089 0.0202
 71: 7.2656 20.331 -1.5186 -1.821 -0.3371 -0.1435 0.0183 0.0021
 72: 1609.9212 1382.898 4.8869 2.7315 -2.5614 -0.3046 0.5194 -0.024
 73: 21.4166 -15.3099 -11.4562 3.4808 1.5131 -0.7695 -0.1291 0.1459
 74: AUGUST
 75: 3 4 11 19 26 0
 76: 1521.7581 1390.0136 -3.1559 -0.4979 1.4662 0.2542 -0.156 -0.0598
 77: -17.945 -17.8421 8.2103 2.4335 -0.4986 -0.131 -0.0506 0.0044
 78: 1422.4886 1398.1519 4.5249 -1.6817 -0.6094 0.3282 -0.0329 -0.0497
 79: -14.1715 18.0313 4.0286 -2.2021 0.2746 0.0386 -0.0702 0.012
 80: 1699.314 1389.5166 -7.9054 0.1184 1.9809 0.5594 -0.2556 -0.1807
 81: 23.0191 11.3155 -8.628 -3.2442 -0.082 0.4967 0.1532 -0.0478
 82: 1589.387 1390.0255 1.9681 -2.5713 -0.1632 0.5191 -0.0639 0.0068
 83: 0.9966 -26.4913 0.2751 4.5876 -0.5062 -0.2009 -0.0985 -0.0132
 84: SEPTEMBER
 85: 3 2 10 18 25 0
 86: 1499.4632 1387.86 3.1033 2.7493 -0.671 -0.6492 0.1419 0.0874
 87: -26.1379 1.863 10.8002 -1.2425 -0.8802 0.229 0.0679 -0.0254
 88: 1772.6156 1401.1619 -1.5756 -2.2934 -0.0367 0.0834 0.0441 0.0103
 89: 4.7582 20.7168 -1.1496 -1.7675 -0.0784 -0.1108 -0.0048 0.0136
 90: 1676.0889 1383.2314 0.8292 3.5993 -1.1083 -0.9428 0.2332 0.1447
 91: 24.3492 -10.183 -12.7724 0.9577 1.7519 -0.0319 -0.1848 0.0247
 92: 1570.462 1386.3477 -2.48 1.005 1.6181 -0.0566 -0.2634 -0.0336
 93: -20.1704 -17.0188 10.4781 2.395 -1.2311 -0.1715 0.0662 0.0262
 94: OCTOBER
 95: 30 1 9 17 24 31
 96: 1492.4707 1394.7533 5.8645 -0.8982 -0.9849 0.3823 0.0469  0.0766
 97: -20.4381 14.2461 6.5493 -2.3402 0.1544 0.1227 -0.0798 -0.0003
 98: 1771.228 1395.7889 -5.648 -1.2751 0.8071 0.3725 0.0058 -0.052
 99: 18.032 16.5073 -5.5514 -2.4786 -0.2008 0.116 0.0737 -0.0066
100: 1667.4704 1388.7175 2.5867 -1.3861 -1.2037 0.4924 0.1126 -0.0845
101: 13.1067 -23.1856 -6.9943 3.7124 0.7437 -0.1571 -0.0051 -0.0111
102: 1561.2979 1384.1831 3.2936 3.9844 -0.7886 -1.0501 0.1816 0.1618
103: -26.5635 -1.5612 12.5639 -1.3169 -1.4108 0.4484 0.1303 -0.0602
104: NOVEMBER
105: 4 8 16 22 30 0
106: 1481.0168 1401.527 0.685 -2.4076 0.0302 0.0506 -0.0196 0.0195
107: -3.2735 21.0512 1.0065 -1.6591 -0.0259 -0.1506 0.0081 0.0218
108: 1751.1928 1385.827 -1.7277 3.7418 0.2249 -1.012 -0.0672 0.1642
109: 26.6394 -0.456 -11.8156 -0.6973 1.2636 0.147 -0.136 -0.0212
110: 1649.4163 1389.353 -4.262 -1.9732 1.2011 0.6879 -0.0315 -0.0883
111: -10.0935 -24.8188 4.4746 4.562 0.0652 -0.3404 -0.1244 -0.0053
112: 1541.8447 1394.6763 7.0573 -1.7195 -1.101 0.6231 -0.0114 -0.0927
113: -19.1307 15.9411 5.8256 -2.7344 0.4998 0.101 -0.1422 0.0256
114: DECEMBER
115: 4 8 15 22 30 0
116: 1481.6577 1399.2582 -4.2324 -2.3219 0.3609 0.2527 0.0877 0.003
117: 10.8023 19.9927 -2.6535 -2.1368 -0.4263 -0.0802 0.0697 0.0302
118: 1740.7656 1387.9731 4.6735 0.5151 -2.0277 0.264 0.2908 -0.1162
119: 20.6523 -16.1923 -9.4108 2.8479 0.6082 -0.4284 0.0436 0.0679
120: 1641.5314 1384.1566 -4.4209 2.9232 2.3458 -0.4248 -0.4711 -0.0073
```

```
121: -23.1024 -13.276 11.5975 2.8067 -1.412 -0.6415 0.1165 0.1218
122: 1551.2149 1398.8825 4.7572 -2.5992 -0.3171 0.29 -0.1149 0.0133
123: -11.63 19.7582 2.8291 -2.1343 0.504 -0.1033 -0.0728 0.023
124:
125:
```

FILE MOONDAT.

Menu File

```
                          MICROFIX:  BEES
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------



                    1. DENSITY ALTITUDE

                    2. MOON PHENOMENA

                    3. SUNRISE/SUNSET/TWILIGHT TIMES
```

FILE BEES.MNU

```
                    MICROFIX BEES:   DENSITY ALTITUDE
-------------------------------------------------------------------------
                          HELICOPTER SELECTION
-------------------------------------------------------------------------

              1. AH-1G                          8. EH-1H

              2. AH-1S                          9. OH-6A

              3. CH-47A                        10. OH-58A

              4. CH-47B                        11. OH-58C

              5. CH-47C                        12. UH-1C/M

              6. CH-54A                        13. UH-1D/H

              7. CH-54B                        14. UH-60A
```

FILE HELI.MNU

1:
2:
3:
4:
5:
6:
7:
8: 1. Moonrise/Moonset times (1984)
9:
10:
11: 2. Moon phase dates for months in 1984
12:
13:
14:
15:
16:

FILE MOON.MNU

Program Listings

```
 1: (* ************************************************************************
 2: *
 3: * PROGRAM BEES
 4: *
 5: * Programmer:       Susan R. Wheeler, Ga Tech/GTRI
 6: * Source file:      BEES.SRC
 7: * Last modified:    7/23/84
 8: *
 9: * This is the driver module for the BEES system. It displays the main menu
10: * and chains to an application or terminates BEES as the user chooses.
11: *
12: ************************************************************************ *)
13:
14: PROGRAM BEES;
15:
16: VAR
17:  OPTION,QUIT: INTEGER;
18:  CHAINFIL: FILE;
19:
20:
21: EXTERNAL FUNCTION CHARTOINTEGER(LINE: STRING): INTEGER;        (* BEESLIB *)
22: EXTERNAL PROCEDURE CLRSCR;                                     (* CURLIB  *)
23: EXTERNAL PROCEDURE GETMENUCHOICE(MX:INTEGER;VAR C:INTEGER);    (* BEESLIB *)
24: EXTERNAL PROCEDURE GOTOXY(X,Y: BYTE);                         (* CURLIB  *)
25: EXTERNAL PROCEDURE MENU(NAME: STRING);                        (* CURLIB  *)
26:
27:
28: BEGIN
29:  QUIT:=0;
30:  CLRSCR;
31:  MENU('BEES.MNU');
32:  GETMENUCHOICE(3,OPTION);
33:  IF OPTION <> QUIT THEN BEGIN
34:    CASE OPTION OF
35:      1: ASSIGN(CHAINFIL,'DENSALT.COM');
36:      2: ASSIGN(CHAINFIL,'MOON.COM');
37:      3: ASSIGN(CHAINFIL,'SUNRISE.COM');
38:     END;  (* CASE *)
39:    CLRSCR;
40:    GOTOXY(1,10);
41:    WRITE('Working.....');
42:    RESET(CHAINFIL);
43:    CHAIN(CHAINFILE)
44:   END  (* IF *)
45: END.  (* BEES *)
46:
47:
```

```
 1: (* ************************************************************************
 2: *
 3: * MODULE IOMOD
 4: *
 5: * Programmer:   Susan R. Wheeler, Ga Tech/GTRI
 6: * Source file:  IOMOD.SRC
 7: * Last revised: 8/31/84
 8: *
 9: * This module contains I/O-related procedures and functions to be part of
10: * the BEESLIB library.
11: *
12: ************************************************************************ *)
13:
14: MODULE  IOMOD;
15:
16:
17: TYPE
18:   STR6=  STRING[6];
19:   STR8=  STRING[8];
20:   STR15= STRING[15];
21:   STR30= STRING[30];
22:   STR35= STRING[35];
23:
24: EXTERNAL FUNCTION @BDOS(FUNC:INTEGER; PARM:INTEGER) : INTEGER;
25: EXTERNAL PROCEDURE CLRLN;
26: EXTERNAL PROCEDURE CLREOS;
27: EXTERNAL PROCEDURE GOTOXY(X,Y: BYTE);
28:
29:
30: FUNCTION CHARTOINTEGER(LINE: STR6): INTEGER;
31:
32: (* convert string of digits in line to an integer *)
33:
34: VAR
35:  IDX,NUMBER: INTEGER;
36:
37: BEGIN
38:   NUMBER:= 0;
39:   FOR IDX:= 1 TO LENGTH(LINE) DO
40:     NUMBER:= NUMBER * 10 + ORD( LINE [IDX] ) - ORD('0');
41:   CHARTOINTEGER:= NUMBER
42: END;
43:
44:
45: FUNCTION CHARTOREAL(LINE: STR8): REAL;
46:
47:   (* convert a string (up to 8 chars to a real number *)
48:   (* does minimal error checking                      *)
49:
50: VAR
51:   IDX,DIVISOR,START: INTEGER;
52:   NUMBER: REAL;
53:   FOUNDPOINT,FOUNDMINUS: BOOLEAN;
54:
55: BEGIN
56:   NUMBER:= 0;
57:   DIVISOR:= 10;
58:   FOUNDPOINT:= FALSE;
59:   FOUNDMINUS:= FALSE;
60:   START:= 0;
```

```
  REPEAT START:= START + 1 UNTIL LINE[START] <> ' '; (* ck leading blanks *)
  IF LINE[START] = '-' THEN BEGIN
     FOUNDMINUS:= TRUE;
     START:= START + 1
    END;
  FOR IDX:= START TO LENGTH(LINE) DO
   IF LINE[IDX] = '.' THEN FOUNDPOINT:= TRUE
   ELSE
     IF FOUNDPOINT THEN BEGIN
        NUMBER:= NUMBER + ((ORD(LINE[IDX])) - (ORD('0'))) / DIVISOR;
        DIVISOR:= DIVISOR * 10
       END
     ELSE NUMBER:= (NUMBER * 10) + (ORD(LINE[IDX]) - (ORD('0')));
  IF FOUNDMINUS THEN NUMBER:= -1 * NUMBER;
  IF ABS(NUMBER) < 1.0E-10 THEN NUMBER:= 0.0;
  CHARTOREAL:= NUMBER
 END;


FUNCTION INTIN(PROMPT:STR30;MIN,MAX: INTEGER): INTEGER;

  (* get an integer input in the interval [MIN,MAX], Repeat PROMPT til *)
  (* they enter a valid input.                                         *)

VAR
   LINE: STRING[5];
   DUMMY: INTEGER;

BEGIN
  REPEAT
    WRITE(PROMPT);
    READLN(LINE);
    DUMMY:= CHARTOINTEGER(LINE)
  UNTIL (DUMMY >= MIN) AND (DUMMY <= MAX);
  INTIN:= DUMMY
END;


FUNCTION KEYPRESSED : BOOLEAN;          (* detect Keyboard input *)

BEGIN
KEYPRESSED := (@BDOS(11,0) <> 0)
END;


FUNCTION REALIN(PROMPT:STR35;MIN,MAX:REAL): REAL;

   (* get a real number in interval [MIN,MAX]. Repeat PROMPT  *)
   (* til CHARTOREAL thinks it has a valid input.             *)

VAR
  LINE: STRING[9];
  NUMOUT: REAL;

BEGIN
 REPEAT
   WRITE(PROMPT);
   READLN(LINE);
   NUMOUT:=CHARTOREAL(LINE);
 UNTIL (NUMOUT <= MAX) AND (NUMOUT >= MIN);
```

```
 REALIN:= NUMOUT
END;


PROCEDURE Get_next_key_pressed( VAR  C : CHAR );

(* most of procedure from larry becker 12/12/84 *)
(* empty keyboard buffer and wait for keypress, return with key value *)

VAR
  K,I : INTEGER;

BEGIN
I := 0;
REPEAT
          (* empty keyboard buffer of type-aheads *)

    K:= @BDOS(6,$FF); (* DIRECT CONSOLE I/O *)
    I := I + 1;
UNTIL I = 128;              (* empty 128 characters ahead *)
REPEAT
  I := @BDOS(6,$FF);
UNTIL I <> 0;
C:= CHR(I);
                            (* xfer to upper case if necessary *)
IF (C >= 'a') AND (C <= 'z') THEN
 BEGIN
  I:= I - (ORD('a') - ORD('A'));
  C:= CHR(I)
 END;
K:= @BDOS(6,$FF);  (* DIRECT CONSOLE I/O *)
END; (* Get_key *)


PROCEDURE GET_A_YESNO_ANSWER(VAR C:CHAR);

               (* get keyboard input limited to (Y,y,N,n) *)
               (* followed by RETURN                     *)
BEGIN
 REPEAT
  GET_NEXT_KEY_PRESSED(C);
  IF (C='Y') OR (C='N') THEN
   BEGIN
    WRITE(C);
    READLN;
    EXIT
   END
  ELSE
    WRITE(CHR(7));
 UNTIL FALSE (* FOREVER *)
END;


PROCEDURE GETMENUCHOICE(MAX:INTEGER; VAR OPTION:INTEGER);

(* returns an integer in interval [0,MAX] in the OPTION argument *)

VAR CHOICE: STRING[3];

BEGIN
```

```
  GOTOXY(1,21);
  CLREOS;
  WRITELN('------------------------------------------------------------------');
  WRITE('Enter a number from 1 to ',MAX:2,' and RETURN:  RETURN to exit. ');
  GOTOXY(39,22);
  READLN(CHOICE);
  IF CHOICE = '' THEN OPTION:= 0
  ELSE OPTION:= CHARTOINTEGER(CHOICE);
  IF (OPTION < 0) OR (OPTION > MAX) THEN REPEAT
    GOTOXY(0,22);
    CLRLN;
    WRITE('Sorry, ',CHOICE, ' is not a valid option. RETURN to continue:');
    READLN;
    GETMENUCHOICE(MAX,OPTION)
   UNTIL (OPTION >= 0) AND (OPTION <= MAX)
END;


PROCEDURE MENU(NAME:STR15);

CONST
   END_FILE      = $1A;              (* displays text file NAME on the *)
                                     (* terminal screen,               *)
VAR
    SRC : TEXT;
    CH : CHAR;
    ERR : INTEGER;


BEGIN (* menu *)
OPEN(SRC,NAME,ERR);
IF ERR=255 THEN
   BEGIN
   WRITELN('NO FILE ',NAME);
   EXIT
   END;
CH:= GNB(SRC);
REPEAT
   ERR:=@BDOS(6,ORD(CH));
   CH:= GNB(SRC);                        (* get next byte from source file *)
UNTIL CH=CHR(END_FILE)
END; (* MENU *)


PROCEDURE PAUSE;                 (* get keyboard input before continuing *)

BEGIN
   GOTOXY(0,23);
   WRITE('RETURN to continue: ');
   READLN
END;


MODEND.


```

```
(* ************************************************************************
*
* MODULE MATHMOD
*
* Programmer:   Susan R. Wheeler, Ga Tech/GTRI
* Source file:  MATHMOD.SRC
* Last revised: 8/22/84
*
* This module contains mathematical procedures and functions to be part of
* the BEESLIB library.
*
************************************************************************ *)

MODULE MATHMOD;       (* MATHMOD.SRC revised 2/28/84 *)

EXTERNAL FUNCTION @BDOS(FUNC: INTEGER; PARM: INTEGER): INTEGER;

FUNCTION ASIN(ARG: REAL) : REAL;          (* arccosin, output is radians *)

BEGIN
 ASIN:= ARCTAN(ARG / SQRT(1 - SQR(ARG)));
END;

FUNCTION ACOS(ARG: REAL) : REAL;          (* arcsin, output is radians *)

BEGIN
 ACOS:= 1.5708 - ARCTAN(ARG / SQRT(1 - SQR(ARG)));
END;

FUNCTION DTR(ARG: REAL) : REAL;           (* convert degrees to radians *)

 BEGIN
  DTR:= 6.283185 * ARG / 360
 END;

FUNCTION RMOD(ARG, MODF: REAL) : REAL;    (* modulus with real arguments *)

VAR TEMP: REAL;

 BEGIN
  TEMP:= ARG - MODF * TRUNC (ARG / MODF);
  IF TEMP < 0 THEN TEMP:= MODF + TEMP;
  RMOD:=TEMP
 END;

FUNCTION RTD(ARG: REAL) : REAL;           (* radians to degrees *)

BEGIN
 RTD:= 360 * ARG / 6.283185
END;

PROCEDURE GETLOCALTIME(LONG: REAL);       (* get absolute time difference *)
                                          (* from the greenwich meridian  *)
VAR
 HCHANGE,MCHANGE: INTEGER;
 MINUTES: REAL;
 CHANGE: STRING;

BEGIN
  IF LONG < 0.0 THEN CHANGE:= 'earlier than'
```

```
61:     ELSE IF LONG > 0.0 THEN CHANGE:= 'later than'
62:       ELSE CHANGE:= 'the same as';
63:   HCHANGE:= TRUNC(ABS(LONG/15));
64:   MINUTES:= ABS(LONG) - 15 * HCHANGE;
65:   MCHANGE:= TRUNC(4 * MINUTES);
66:   WRITELN;
67:   WRITE('Time at this longitude is ',CHANGE,' ZULU time');
68:   IF LONG <> 0 THEN WRITELN(' by ',HCHANGE:2,' hours ',
69:      MCHANGE:2,' minutes')
70: END;
71: MODEND.
72:
```

```
(* ************************************************************************
*
* MODULE DENSALT        (source file = DENSALT.SRC)
*
* Programmer:   Susan R. Wheeler, Georgia Tech/GTRI
* Source File:  DENSALT.SRC
* Last modified:   8/31/84
*
*   This is the main module of the program which computes density
* altitude from temperature, elevation, and pressure inputs and
* optionally outputs load capacities for certain helicopter models
* at the density altitude just computed. 14 helicopter models have
* been implemented. Model choice is made from a menu.
*    All possible comments have been preserved from the original BASIC
* source as well as most variable names. Floating point precision of the
* MT+ PASCAL compiler is approximately 6.5 digits whereas the precision
* of BASIC on the HP85 was 12 digits. All digits appearing in the BASIC
* program were transferred to the PASCAL version, however. Comparison
* of load capacities output by the PASCAL and BASIC versions shows a
* typical agreement within 3-5 pounds. No test case so far has varied more
* than 8 pounds.
*    DENSITY ALTITUDE is chained to by, and chains back to the main BEES menu.
*
************************************************************************ *)

PROGRAM DENSALT;


TYPE STR8 = STRING[8];
     STR30 = STRING[30];
     STR35 = STRING[35];

  VAR

   ANSWER: CHAR;
   BP,C,CDEWPT,D,DA,D2,E,ELEV,F,FDEWPT,G,H,I: REAL;
   ICEPT,PA,R,S,STDATM,STEAMPT,SVP,TEMP,Z: REAL;
   CHAINFIL: FILE;

EXTERNAL FUNCTION REALIN(PROMPT:STR30;MAX,MIN:REAL):REAL;   (* BEESLIB *)
EXTERNAL PROCEDURE CLRSCR;                                  (* CURLIB  *)
EXTERNAL PROCEDURE DOLOADCAPACITYCALCULATIONS;              (* COPTER  *)
EXTERNAL PROCEDURE GETAYESNOANSWER(VAR C: CHAR);            (* BEESLIB *)
EXTERNAL PROCEDURE GETMENUCHOICE(MX:INTEGER;VAR C:INTEGER); (* BEESLIB *)
EXTERNAL PROCEDURE MENU(NAME:STRING);                       (* CURLIB  *)
EXTERNAL PROCEDURE PAUSE;                                   (* BEESLIB *)


PROCEDURE CALCULATEDENSITYALTITUDE;

 BEGIN
   IF BP <> 0 THEN                 (* CONVERT PRESSURE TO PRESSURE ALTITUDE *)
     GETPALTITUDEFROMPRESSURE
   ELSE                           (* CONVERT PRESSURE ALTITUDE TO PRESSURE *)
     GETPRESSUREFROMPALTITUDE;
   ICEPT:= 273.16;
   STDATM:= 1013.246;
   STEAMPT:= 373.16;
   IF CDEWPT > 0 THEN
      BEGIN
```

```
61:         CDEWPT:= CDEWPT + ICEPT;
62:         D:= -7.90298 * (STEAMPT / CDEWPT - 1);
63:         E:= 5.02808 * LN(STEAMPT / CDEWPT) / LN(10);
64:         R:= 1 - CDEWPT / STEAMPT;
65:         F:= 1.3816E-7 * (EXP(11.344 * R * LN(10)) - 1);
66:         S:= STEAMPT / CDEWPT - 1;
67:         G:= 8.1328E-3 * (EXP(-3.49149 * S * LN(10)) - 1);
68:         H:= LN(STDATM) / LN(10);
69:         I:= D + E - F + G + H;
70:         Z:= EXP(I * LN(10));
71:         SVP:= Z * 0.02953
72:       END
73:    ELSE
74:       BEGIN
75:         D2:= CDEWPT + ICEPT;
76:         D:= -9.09718 * (ICEPT / D2 - 1);
77:         E:= -3.56654 * LN(ICEPT / D2) / LN(10);
78:         F:= 0.876793 * (1-D2) / ICEPT;
79:         G:= LN(6.1071) / LN(10);
80:         I:= D + E + F + G;
81:         Z:= EXP(I * LN(10));
82:         SVP:= Z * 0.02953
83:       END;
84:    F:= 0.622 * SVP / (BP-SVP);  {* MXING RATIO CALCULATIONS *}
85:    I:= (TEMP+459.69) * (1+1.61 * F) / (1+F); {* VIRTUAL TEMP CALC *}
86:    DA:= (1 - EXP(0.235 * LN(17.326 * BP / I))) * 145366.0; {*DENSITY ALT*}
87: END;
88:
89:
90: PROCEDURE DODENSITYALTITUDECALCULATIONS;
91:
92: BEGIN
93:   GETDENSITYALTITUDEINPUTS;
94:   CALCULATEDENSITYALTITUDE;
95:   PRINTDENSITYALTITUDE;
96:   WRITELN;WRITELN('Do you want load capacities for selected helicopters');
97:   WRITE('      at this density altitude? (Y or N) ');
98:   GETAYESNOANSWER(ANSWER);
99:   IF ANSWER = 'Y' THEN DOLOADCAPACITYCALCULATIONS
100: END;
101:
102:
103: PROCEDURE HEADING;
104:
105:  BEGIN
106:   CLRSCR;
107:   WRITELN('                    MICROFIX BEES:     DENSITY ALTITUDE');
108:   WRITELN
109:   ('-------------------------------------------------------------------',
110:   '-----');
111:   WRITELN
112:   ('-------------------------------------------------------------------',
113:   '-----')
114:  END;
115:
116:
117: PROCEDURE GETDENSITYALTITUDEINPUTS;
118:
119:  BEGIN
120:    HEADING;
```

```
121:    WRITELN;
122:    ELEV:= REALIN('Enter STATION ELEVATION (feet): ',-500,30000.0);
123:    WRITELN;
124:    TEMP:=REALIN('Enter TEMPERATURE (degrees F): ',-150,150);
125:    WRITELN;
126:    FDEWPT:= REALIN('Enter DEW POINT (degrees F): ',-150,150);
127:    CDEWPT:=5 * (FDEWPT-32) / 9;
128:    WRITELN;
129:    WRITELN('Enter UNCORRECTED STATION PRESSURE (inches Hg) ');
130:    WRITELN('   or a "0" if only the PRESSURE ALTITUDE is known: ');
131:    WRITELN;
132:    BP:= REALIN('Enter 0 or barometric pressure: ',0,33);
133:    WRITELN
134:  END;
135:
136:
137:
138: PROCEDURE GETPALTITUDEFROMPRESSURE;
139:
140:                    {* CONVERT PRESSURE TO PRESSURE ALTITUDE *}
141:
142:      BEGIN
143:        PA:= BP * 33.8639;  {* CONVERT INCHES HG TO MILLIBARS *}
144:        IF PA >= 1013.25 THEN  PA:= 26859.7345224 - 26.5185 * PA
145:        ELSE
146:          IF PA < 705.09 THEN  PA:= 50125.7923315 - 79.0246642817 * PA
147:                                  + 0.030808325969 * PA * PA
148:          ELSE PA:= 42648.3525204 - 57.4057011468 * PA
149:                                  + 1.51260276686E-2 * PA * PA;
150:        IF PA < 0 THEN C:= 7.13966666655 + 0.8264698 * PA + 1.35663E-5*PA*PA
151:        ELSE
152:          IF PA < 5000 THEN C:= -11.4763604151 + 1.40661336336E-2 * PA
153:                               - 2.36456659055E-6 * PA * PA
154:          ELSE
155:            IF PA < 9000 THEN C:= -1175.00071 + 0.73075 * PA
156:                   - 1.65866E-4 * PA * PA + 1.6224E-8 * PA * PA * PA
157:                   -5.78465E-13 *PA * PA * PA * PA
158:            ELSE
159:              IF (PA > 9350) AND (PA < 9700) THEN
160:                  C:= 1.18740544813E+5 - 38.10748 * PA + 4.0738E-3 * PA * PA
161:                      - 1.45063E-7 * PA * PA * PA
162:              ELSE
163:                IF PA >= 9700 THEN
164:                  C:= -1.27373891E+3 + 0.2686 * PA - 1.83266E-5 * PA * PA
165:                      + 4.05324E-10 * PA * PA * PA;
166:        PA:= PA+C
167:      END;  {* BP <> 0 *}
168:
169:
170: PROCEDURE GETPRESSUREFROMPALTITUDE;
171:
172:                  {*  CONVERT PRESSURE ALTITUDE TO PRESSURE *}
173:
174:      BEGIN
175:        WRITELN;
176:        PA:= REALIN('Pressure Altitude (feet)? ',-1000.0,10000.0);
177:        BP:= 1013.08968413 - 3.60310856232E-2 * PA + 4.38653185363E-7*PA*PA;
178:        C:= -0.1403422 + 5.58757E-4 * PA - 9.1921E-8 * PA * PA
179:             + 3.46176E-12 * PA * PA * PA;
180:        BP:= BP-C;
```

```
181:        BP:= BP / 33.8639      {* CONVERT MILLIBARS TO INCHES HG *}
182:      END;  {* BP = 0 *}
183:
184:
185: PROCEDURE PRINTDENSITYALTITUDE;
186:
187: BEGIN
188:    HEADING;
189:    WRITELN;WRITELN;
190:    WRITE('           ');
191:    WRITE ('Station Elevation = ',ELEV:6:0); WRITELN (' feet');
192:    WRITE('           ');
193:    WRITE ('Temperature       = ',TEMP:6:0); WRITELN (' degrees Farenheit');
194:    WRITE('           ');
195:    WRITE ('Pressure          = ',BP:6:2); WRITELN (' inches of mercury');
196:    WRITE('           ');
197:    WRITE ('Pressure Altitude = ',PA:6:2); WRITELN (' feet');
198:    WRITELN; WRITE ('           ',
199:                    'Your Density Altitude is ',DA:6:2); WRITELN (' feet');
200: END;
201:
202:
203:                    (* MAIN PROCEDURE *)
204:
205: BEGIN
206:  REPEAT
207:    DODENSITYALTITUDECALCULATIONS;
208:    WRITELN;WRITE('Want to calculate another density altitude? (Y or N) ');
209:    GETAYESNOANSWER(ANSWER)
210:  UNTIL ANSWER = 'N';
211:  ASSIGN(CHAINFIL,'BEES.COM');    (*  chain back to main BEES menu *)
212:  RESET(CHAINFIL);
213:  CHAIN(CHAINFIL)
214: END.
```

```
(* ************************************************************************
*
* MODULE COPTER
*
* Programmer:    Susan R. Wheeler, Ga Tech/GTRI
* Source file:   COPTER.SRC
* Last modified:  7/24/84
*
* This module contains procedures to handle the helicopter model menu used
* by the Density Altitude program, and procedures to perform load capacity
* calculations for some of the helicopters on the menu. The rest of the
* helicopters had to be put into another file (NEWCOP.SRC) because this file
* exceeded the size limit of the SPP editor.
*
* The helicopter load capacity equations and parameters in this source file
* and in NEWCOP.SRC were extracted from the corresponding ETL BEES density
* altitude program.
************************************************************************ *)

MODULE COPTER;          (* revised 2/28/84 *)

VAR
 TEMP,PA,DA: EXTERNAL REAL;
 WT,MAXWT,GROSSWT,A,B,TORQUE,MAXTORQUE: REAL;
 Y,YSCALE,TEMP2,TEMP3,TEMP4,DA2,DA3,DA4: REAL;
 CRAFT,OIL,CREW: REAL;
 NCREW,HELNUM: INTEGER;
 HELTYPE: STRING;

EXTERNAL PROCEDURE GETMENUCHOICE(MX:INTEGER;VAR C:INTEGER); (* BEESLIB *)
EXTERNAL PROCEDURE MENU(NAME:STRING);                       (* CURLIB  *)
EXTERNAL PROCEDURE PAUSE;                                   (* BEESLIB *)
EXTERNAL PROCEDURE CLRSCR;                                  (* CURLIB  *)
EXTERNAL PROCEDURE HEADING;                                 (* DENSALT *)
EXTERNAL PROCEDURE CH54A;                                   (* NEWCOP  *)
EXTERNAL PROCEDURE CH54B;                                   (*   "     *)
EXTERNAL PROCEDURE OH6A;                                    (*   "     *)
EXTERNAL PROCEDURE OH58A;                                   (*   "     *)
EXTERNAL PROCEDURE OH58C;                                   (*   "     *)
EXTERNAL PROCEDURE UH1CM;                                   (*   "     *)
EXTERNAL PROCEDURE UH1DH;                                   (*   "     *)
EXTERNAL PROCEDURE UH60A;                                   (*   "     *)

PROCEDURE SETUP_LOADCAPACITYCALCULATIONS;
 BEGIN
   TEMP:= (TEMP - 32) * 5 / 9;   {* convert to centigrade *}
   TEMP2:= SQR (TEMP);
   TEMP3:= TEMP * TEMP2;
   TEMP4:= SQR (TEMP2);
   DA2:= SQR(DA);
   DA3:= DA * SQR(DA);
   DA4:= SQR(DA2);
 END;

PROCEDURE COMPUTE_ALOADCAPACITY;

BEGIN
  CLRSCR;
  MENU('HELI.MNU');      (* File HELI.MNU contains text of helicopter menu *)
```

```
  GETMENUCHOICE(14,HELNUM); (* 14 is max valid menu option number *)
  CASE HELNUM OF
    0: EXIT;                (* 0 is 'quit the menu' *)
    1: AH_1G;
    2: AH_1S;
    3: CH_47A;              (* Procedures containing calculations for the *)
    4: CH_47B;              (* helicopters indicated                      *)
    5: CH_47C;
    6: CH54A;
    7: CH54B;
    8: UH1DH;
    9: OH6A;
    10: OH58A;
    11: OH58C;
    12: UH1CM;
    13: UH1DH;
    14: UH60A;
  END   (* CASE *)
END;    (* load capacity *)


PROCEDURE DOLOADCAPACITYCALCULATIONS;

BEGIN
  SETUPLOADCAPACITYCALCULATIONS;
  REPEAT COMPUTEALOADCAPACITY UNTIL HELNUM=0
END;


PROCEDURE PRINT_OUTPUT;
 BEGIN
  TEMP:= TRUNC (TEMP * 100) / 100;
  HEADING;
  WRITELN;WRITELN('                          ',
                  'HELICOPTER LOADING CAPABILITY');
  WRITELN         ('                          ',
                   '----------------------------');WRITELN;
  WRITELN('               Helicopter type = ',HELTYPE);
  WRITELN('                           OAT = ',TEMP:8:2,' degrees Centigrade');
  WRITELN('              Pressure Altitude = ',PA:8:2,' feet');
  WRITELN('               Density Altitude = ',DA:8:2,' feet');
  WRITELN;WRITELN;
  WRITELN('Maximum allowable load (cargo, fuel, passenger mix) = ',
     TRUNC(WT):6:0,' lbs   (approx)');
  WRITELN;
  WRITELN('Allowable load assumes:');WRITELN;
  WRITELN('         Aircraft basic weight = ',TRUNC(CRAFT):10:0,' lbs');
  WRITELN('                           Oil = ',TRUNC(OIL):10:0,' lbs');
  NCREW:=TRUNC(CREW);
  CREW:=CREW * 200;
  WRITELN('                      Crew(',NCREW:1,') = ',
          TRUNC(CREW):10:0,' lbs');
  PAUSE
 END;


PROCEDURE AH_1G;
 BEGIN
  HELTYPE:= 'AH-1G (T53-L-13B)';
  CRAFT:=6067;
```

```
121:   OIL:=24;
122:   CREW:=1;
123:   MAXWT:=9500;
124:   MAXTORQUE:=50;   {* PSI *}
125:   A:= 98872.516782 - 177.53792 * TEMP + 3.058016 * TEMP2
126:        +0.1009999 * TEMP3 - 4.65033E-3 * TEMP4;
127:   B:= -23549.2176 - 5.53587 * TEMP - 1.18097 * TEMP2
128:       -0.0315638 * TEMP3 +1.35519E-3 * TEMP4;
129:   TORQUE:=EXP((PA-A) / B);
130:   IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
131:   YSCALE:=0.2 * TORQUE - 5;
132:   A:= 5539.57727 - 4.67087E-3 * DA - 8.47021E-7 * SQR(DA);
133:   B:= 812.884 - 0.014 * DA;
134:   GROSSWT:=A + B * YSCALE;
135:   IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
136:   WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
137:   PRINT_OUTPUT
138:
139:  END;
140: PROCEDURE AH_1S;
141:
142:  BEGIN
143:    HELTYPE:= 'AH-1S (T53-L-703), TWO ENGINE OPERATION';
144:    CRAFT:= 6910;
145:    OIL:= 29;
146:    CREW:= 1;
147:    MAXWT:= 10000;
148:    MAXTORQUE:= 56;  {* PSI *}
149:    A:= 85128.094 - 391.18 * TEMP + 6.933 * TEMP2 - 0.0974 * TEMP3;
150:    B:= -18754.672 + 78.948 * TEMP - 2.1278 * TEMP2 - 0.02847 * TEMP3
151:        + 9.4623E-4 * TEMP4;
152:    TORQUE:= EXP ((PA-A) / B);
153:    IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
154:    YSCALE:= 0.2 * TORQUE - 5;
155:    A:= 5587.655 + 4.856E-4 * DA + 1.18276E-6 * DA2 - 9.194E-11 * DA3;
156:    B:=797.3967 - 0.01379 * DA - 6.807E-7 * DA2 + 2.86E-11 * DA3;
157:    GROSSWT:= A + B * YSCALE;
158:    IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
159:    WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
160:    PRINT_OUTPUT
161:
162:  END;
163:
164: PROCEDURE CH_47A;
165:
166: BEGIN
167:
168:   HELTYPE:='CH-47A (T55-L-7/7B), TWO ENGINE OPERATION';
169:   CRAFT:= 17105;
170:   OIL:=28;
171:   CREW:=2;
172:   MAXWT:=33000.0;
173:   MAXTORQUE:=860; {* LB FT *}
174:   A:= 180909.633 - 100.4284 * TEMP - 9.0486 * TEMP2
175:       + 0.2776 * TEMP3 -2.6125E-3 * TEMP4;
176:   B:= -26178.3089 - 9.0711 * TEMP + 1.25465 * TEMP2 - 0.04209 * TEMP3
177:       + 3.8448E-4 * TEMP4;
178:   TORQUE:= EXP ((PA - A)/B);
179:   IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
180:   YSCALE:= 0.02 * TORQUE - 6;
```

```
181:   A:= 17865.134 - 0.03225 * DA + 6.6797E-6 * DA2 - 8.496E-10 * DA3;
182:   B:= 2029.0 - 0.0399 * DA - 2.127E-6 * DA2 + 2.382E-10 * DA3;
183:   GROSSWT:= A + B * YSCALE;
184:   IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
185:   WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
186:   PRINT_OUTPUT
187: END;
188:
189: PROCEDURE CH_47B;
190:
191: BEGIN
192:   HELTYPE:= 'CH47B (T55-L-7B), TWO ENGINE OPERATION';
193:   CRAFT:= 19410;
194:   OIL:= 28;
195:   CREW:= 2;
196:   MAXWT:= 40000.0;
197:   MAXTORQUE:= 860;(* LB FT *)
198:   A:= 180909.633 - 100.4284 * TEMP -9.0486 * TEMP2 + 0.2776 * TEMP3
199:        - 2.6125E-3 * TEMP4;
200:   B:= - 26178.3089 - 9.0711 * TEMP + 1.25465 * TEMP2 - 0.04209 * TEMP3
201:            + 3.8448E-4 * TEMP4;
202:   TORQUE:= EXP ((PA-A) / B);
203:   IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
204:   YSCALE:= 0.01 * TORQUE - 3;
205:   IF TEMP < 0 THEN TEMP:= ABS(TEMP);
206:   IF TEMP > 10 THEN BEGIN
207:      A:= 0;
208:      B:= 1
209:     END
210:   ELSE BEGIN
211:     A:= -0.016543 + 6.9857E-4 * TEMP + 9.64286E-5 * TEMP2;
212:     B:= 0.98997 + 6.02381E-4 * TEMP + 1.42857E-5 * TEMP2 + 2.8333E-6 * TEMP3
213:    END;
214:   Y:= A + B * YSCALE;
215:   A:= 18600.2389 - 4.61E-3 * DA - 4.895E-6 * DA2;
216:   B:= 3938.1889 - 0.07542786 * DA + 1.1847E-6 * DA2;
217:   GROSSWT:= A + B * Y;
218:   IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
219:   WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
220:   PRINT_OUTPUT
221:
222: END;
223: PROCEDURE CH_47C;
224:
225: BEGIN
226:  HELTYPE:= 'CH-47C (T55-L-7C), TWO ENGINE OPERATION';
227:  CRAFT:= 21791;
228:  OIL:= 28;
229:  CREW:= 2;
230:  MAXWT:=40000.0;
231:  MAXTORQUE:=990; (* LB FT *)
232:  A:= 1012.8261 - 0.039624 * PA + 6.5447E-7 * SQR(PA);
233:  B:= - 7.22367 + 3.429E-4 * PA + 7.6667E-9 * SQR(PA);
234:  TORQUE:= A + B * TEMP;
235:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
236:  YSCALE:= 0.0078 * TORQUE - 3.12;
237:  A:= -0.042277 - 4.1146E-4 * TEMP +9.07749E-5 * TEMP2 -1.3535E-6 * TEMP3;
238:  B:= 1.06055 - 3.278E-3 * TEMP + 4.3958E-5 * TEMP2;
239:  Y:= (YSCALE - A)/B;
240:  A:= 22070.8728 - 0.0336509 * DA - 1.70867E-6 * DA2 + 3.01605E-10 * DA3;
```

```
241:  B:= 4997.193 - 0.114 * DA;
242:  GROSSWT:= A + B * Y;
243:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
244:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
245:  PRINT_OUTPUT
246: END;
247: MODEND.
```

```
 1: D* ************************************************************************
 2: *
 3: * MODULE COP2
 4: *
 5: * Programmer:       Susan R. Wheeler, Ga Tech/GTRI
 6: * Source file:      COP2.SRC
 7: * Last modified:    6/29/84
 8: *
 9: * This module contains helicopter load capacity calculations used by the
10: * Density Altitude program. The helicopters in this file would not fit in
11: * file COPTER.SRC because it got too long for the SPP editor to handle.
12: *
13: ************************************************************************ *)
14:
15: MODULE COP2;
16:
17: VAR
18:  C: REAL;
19:  TEMP,PA,DA: EXTERNAL REAL;
20:  WT,MAXWT,GROSSWT,A,B,TORQUE,MAXTORQUE: EXTERNAL REAL;
21:  Y,YSCALE,TEMP2,TEMP3,TEMP4,DA2,DA3,DA4: EXTERNAL REAL;
22:  CRAFT,OIL,CREW: EXTERNAL REAL;
23:  HELNUM: EXTERNAL INTEGER;
24:  HELTYPE: EXTERNAL STRING;
25:
26: EXTERNAL PROCEDURE PRINT_OUTPUT;
27:
28: PROCEDURE CH_54A;
29: BEGIN
30:  HELTYPE:='CH-54A (T73-P-1), TWO ENGINE OPERATION';
31:  CRAFT:= 21895;
32:  OIL:=15;
33:  CREW:=2;
34:  MAXWT:=42000.0;
35:  MAXTORQUE:= 81.5;   (* PERCENT *)
36:  IF TEMP <= - 11 THEN BEGIN
37:     A:=35613.1689 - 1.22378 * TEMP+ 0.56865 * TEMP2;
38:     B:= -296.125 - 0.166 * TEMP - 7.698E-3 * TEMP2
39:    END
40:  ELSE BEGIN
41:    A:=34787.769488 - 70.79067 * TEMP + 0.7604 * TEMP2;
42:    B:= -296.62895 - 0.2883 * TEMP - 0.03579 * TEMP2
43:   END;
44:  TORQUE:=(PA-A)/B;
45:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
46:  YSCALE:= 0.05 * TORQUE - 1;
47:  A:= 0.98398 + 4.568E-4 * TEMP - 2.9856E-6 * TEMP2;
48:  B:= 0.935 + 1.2618E-3 * TEMP;
49:  Y:= A * EXP (B * LN(YSCALE));
50:  A:= 25925.7756 - 0.167786 * DA - 6.1351E-6 * DA2;
51:  B:= 0.4199 - 7.528E-7 * DA -5.4282E-11 * DA2;
52:  GROSSWT:= A * EXP(B * LN(Y));
53:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
54:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
55:  PRINT_OUTPUT
56: END;
57:
58: PROCEDURE CH_54B;
59:
60: BEGIN
```

```
 61:  CRAFT:= 22625;
 62:  OIL:= 15;
 63:  CREW:=2;
 64:  HELTYPE:= 'CH-54B (T73-P-700), TWO ENGINE OPERATION';
 65:  MAXWT:=47000.0;
 66:  MAXTORQUE:=100.0;(* PERCENT *)
 67:  IF TEMP >= -14 THEN BEGIN
 68:    A:= 121998.47092 - 95.3369 * TEMP - 7.31065 * TEMP2;
 69:    B:= -26009.6487 - 30.9293 * TEMP + 0.84051 * TEMP2
 70:        + 0.102388 * TEMP3 - 1.72256E-3 * TEMP4;
 71:   END
 72:  ELSE BEGIN
 73:    A:= 94594.5894 - 3884.3305 * TEMP - 184.65234* TEMP2
 74:        -3.58638 * TEMP3 - 0.02472 * TEMP4;
 75:    B:= -21752.01913 + 473.1486 * TEMP + 16.41736 * TEMP2 + 0.17233 * TEMP3;
 76:   END;
 77:  TORQUE:=EXP((PA - A) / B);
 78:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
 79:  YSCALE:= 0.05 * TORQUE - 1;
 80:  A:= 0.9892857 + 4.166667E-4 * TEMP - 1.382857E-5 * TEMP2 +1.9733E-7 * TEMP3;
 81:  B:= 0.9479 + 1.71E-3 * TEMP - 1.3371E-5 * TEMP2;.
 82:  Y:= A * EXP(B * LN (YSCALE));
 83:  A:= 16542.1559 + 0.1092 * DA - 2.03965E-6 * DA2 - 6.7E-10 * DA3;
 84:  B:= 14187.787 - 0.3305 * DA - 9.9711E-6 * DA2 + 1.2315E-9 * DA3;
 85:  C:= -1253.3072 + 0.03978 * DA + 1.0865E-6 * DA2 - 2.325E-10 * DA3;
 86:  GROSSWT:= A + B * Y + C * SQR(Y);
 87:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
 88:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
 89:  PRINT_OUTPUT
 90: END;
 91:
 92: PROCEDURE OH_6A;
 93:
 94: BEGIN
 95:  CRAFT:= 1050;
 96:  OIL:= 6;
 97:  CREW:= 1;
 98:  HELTYPE:= 'OH-6A (T63-A-5A/-700)';
 99:  MAXWT:= 2550;
100:  MAXTORQUE:= 80.3;(* PSIG *)
101:  A:= 102996.723 - 304.236 * TEMP;
102:  B:= -21965.0879998 + 34.059849213 * TEMP -0.19067595237 * TEMP2
103:      - 1.88977778529E-3 * TEMP3;
104:  TORQUE:= EXP((PA - A) / B);
105:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
106:  YSCALE:= 0.2 * TORQUE - 8;
107:  A:= 1587.6971 - 0.016613 * DA + 4.7096E-7 * DA2;
108:  B:= 134.8292 - 6.686E-5 * DA - 7.266E-8 * DA2;
109:  GROSSWT:= A + B * YSCALE;
110:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
111:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
112:  PRINT_OUTPUT
113: END;
114:
115: PROCEDURE OH_58A;
116:
117: BEGIN
118:  CRAFT:= 1725;
119:  OIL:= 13;
120:  CREW:= 1;
```

```
121:  HELTYPE:= 'OH-58A (T63-A-700)';
122:  MAXWT:= 3000;
123:  MAXTORQUE:= 78; (* PSI *)
124:  A:= 100574.314 - 335.296 * TEMP;
125:  B:= - 21976.59 + 33.793 * TEMP;
126:  TORQUE:= EXP((PA-A) / B);
127:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
128:  YSCALE:= 0.2 * TORQUE - 7;
129:  A:= 1656.74286 - 4.4852E-4 * DA - 2.14579E-7 * DA2;
130:  B:= 179.227 - 0.003 * DA;
131:  GROSSWT:= A + B * YSCALE;
132:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
133:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
134:  PRINT_OUTPUT
135: END;
136:
137: PROCEDURE OH_58C;
138:
139: BEGIN
140:  CRAFT:= 1898;
141:  OIL:= 11;
142:  CREW:= 1;
143:  HELTYPE:= 'OH-58C (T63 A-720)';
144:  MAXWT:= 3200;
145:  MAXTORQUE:= 85; (* PERCENT *)
146:  A:= 111904.6343 - 333.287 * TEMP - 0.9386 * TEMP2;
147:  B:= -22630.94 + 50.807 * TEMP;
148:  TORQUE:= EXP((PA-A) / B);
149:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
150:  YSCALE:= 0.1 * TORQUE - 5;
151:  A:= 2012.5 - 5.05E-3 * DA - 1.05322E-6 * DA2 + 2.2958E-10 * DA3
152:       - 8.3415E-15 * DA4;
153:  B:= 367.6616 - 5.8745E-3 * DA - 4.7198E-7 * DA2;
154:  GROSSWT:= A + B * YSCALE;
155:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
156:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
157:  PRINT_OUTPUT
158: END;
159:
160: PROCEDURE UH_1CM;
161:
162: BEGIN
163:  CRAFT:= 5156;
164:  OIL:= 28;
165:  CREW:= 1;
166:  HELTYPE:= 'UH-1C/M (T53-L-13)';
167:  MAXWT:= 9500;
168:  MAXTORQUE:= 50; (* PSI *)
169:  A:= 104798.2095 - 230.8065 * TEMP - 1.3629 * TEMP2;
170:  B:= -24682.61 - 2.0857 * TEMP - 0.2726 * TEMP2 + 0.0108 * TEMP3;
171:  TORQUE:= EXP  ((PA-A) / B);
172:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
173:  YSCALE:= 0.4 * TORQUE - 12;
174:  A:= 6084.1843 + 0.02774 * DA - 2.473E-6 * DA2;
175:  B:= 437.239 - 0.018 * DA;
176:  GROSSWT:= A + B * YSCALE;
177:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
178:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
179:  PRINT_OUTPUT
180: END;
```

```
181:
182: PROCEDURE UH_1DH;
183:
184: BEGIN
185:  CRAFT:= 5320;
186:  OIL:= 27;
187:  CREW:= 1;
188:  MAXWT:= 9500;
189:  MAXTORQUE:= 50; (* PSI *)
190:  IF HELNUM = 8 THEN HELTYPE:= 'EH-1H (T53-L-13)'
191:  ELSE HELTYPE:= 'UH-1H (T53-L-13)';
192:  A:= 101054.187157 - 18.262435 * TEMP -6.0301 * TEMP2 + 0.14874 * TEMP3
193:      - 2.0619E-3 * TEMP4;
194:  B:= -24007.94261 - 48.213595 * TEMP + 1.35253 * TEMP2 - 0.0387325 * TEMP3
195:      + 4.4448E-4 * TEMP4;
196:  TORQUE:= EXP((PA-A) / B);
197:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
198:  YSCALE:= 0.2 * TORQUE - 4;
199:  A:= 4876.42675 + 9.1772E-3 * DA + 4.2057E-7 * DA2 - 3.2717E-12 * DA3
200:      - 3.484E-15 * DA4;
201:  B:= 861.8876 - 0.0145625 * DA - 2.8954E-7 * DA2;
202:  GROSSWT:= A + B * YSCALE;
203:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
204:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
205:  PRINT_OUTPUT
206: END;
207:
208: PROCEDURE UH_60A;
209:
210: BEGIN
211:  CRAFT:=10984;
212:  OIL:=15;
213:  CREW:= 2;
214:  HELTYPE:= 'UH-60A (T700), TWO ENGINE OPERATION';
215:  MAXWT:= 20250;
216:  MAXTORQUE:= 100; (* PERCENT *)
217:  IF ((TEMP (= -9) AND (PA ( 15000)) OR ((TEMP (= -14) AND (PA ( 15000))
218:    THEN BEGIN
219:     A:= 133092.5625 + 646.745925 * TEMP + 8.2227 * TEMP2;
220:     B:= -27493.0 - 116.449945 * TEMP - 1.5191325 * TEMP2;
221:    END
222:  ELSE BEGIN
223:    A:= 114366.99834 - 166.684866 * TEMP - 28.12099 * TEMP2
224:        + 1.113862 * TEMP3 - 0.011 * TEMP4;
225:    B:= -23803.01534 + 11.427755 * TEMP + 6.567375 * TEMP2
226:        - 0.27295 * TEMP3 + 2.70916E-3 * TEMP4;
227:   END;
228:  TORQUE:= EXP((PA-A) / B);
229:  IF TORQUE > MAXTORQUE THEN TORQUE:= MAXTORQUE;
230:  YSCALE:= 0.1 * TORQUE - 4;
231:  A:= 0.09711 - 1.425E-3 * TEMP - 9.7213E-6 * TEMP2;
232:  B:= 0.936644 + 1.2226448E-3 * TEMP - 3.10881E-6 * TEMP2;
233:  Y:= A + B * YSCALE;
234:  A:= 10745.56338 + 0.0674077 * DA - 1.53675E-7 * DA2 - 2.2571E-10 * DA3;
235:  B:= 1908.70357 - 0.06425 * DA + 1.0E-6 * DA2;
236:  GROSSWT:= A + B * Y;
237:  IF GROSSWT > MAXWT THEN GROSSWT:= MAXWT;
238:  WT:= GROSSWT - (CRAFT + OIL + CREW * 200);
239:  PRINT_OUTPUT
240: END;
```

```
241: MODEND.
242:
```

```
 1: (* ***************************************************************
 2: *
 3: * PROGRAM RANREAD
 4: *
 5: * Programmer:       Susan R. Wheeler, Ga Tech/GTRI
 6: * Source file:     RANREAD.SRC
 7: * Last modified:   8/31/84
 8: *
 9: * This program reads sequential disk file MOONDAT and writes random
10: * access disk file MOONRAN, which is used by the Moon Phenomena
11: * program. These files are further discussed in the programmer
12: * reference documentation for the Moon Phenomena program.
13: *
14: *************************************************************** *)
15:
16: PROGRAM RANREAD;
17:
18: TYPE
19:  MOONDATA = RECORD                                  (* record structure *)
20:           MONTHNAME: STRING[9];                     (* for MOONRAN      *)
21:           YEAR: INTEGER;
22:           RA: INTEGER;                              (* 'A' constant     *)
23:           RD5: REAL;                                (* 'D5' constant    *)
24:           WD: INTEGER;
25:           PHASEDATE:  ARRAY[1..5] OF INTEGER;
26:           MONTHDATA:  ARRAY[1..4,1..16] OF REAL;;
27:         END;
28:
29: VAR
30:
31:  RF: FILE OF MOONDATA;
32:  ERR,II: INTEGER;
33:
34: PROCEDURE MOVEDATATORANDOMFILE(AFILE: STRING;RFILE: STRING);
35:
36: CONST
37:   END_FILE     = $1A;
38:
39: VAR
40:    SRC : TEXT;
41:    MONTH,WEEK,K: INTEGER;
42:
43: BEGIN
44:  OPEN(SRC,AFILE,ERR);                    (* open sequential file *)
45:  IF ERR=255 THEN
46:   BEGIN
47:   WRITELN('ASCII input file ',AFILE,' does not exist.');
48:   EXIT
49:   END;
50:  WRITELN('Data will be moved from ASCII file ',AFILE,' to random file ',
51:           RFILE);WRITELN;
52:  OPEN(RF,RFILE,ERR);                     (* open random file *)
53:
54:       (* Read data from the sequential file into the MOONDATA record. *)
55:       (* The record with its current contents is written to specified *)
56:       (* record locations in the random access output file.           *)
57:       (*    The RF^ prefix indicates an element of the record.       *)
58:
59:  READLN(SRC,RF^.YEAR);       (* These values appears just once in the  *)
60:  READLN(SRC,RF^.RA);         (* sequential file but will be in every   *)
```

```
 61:  READLN(SRC,RF^.RD5);       (* record of the random file.            *)
 62:
 63:      (* Read data for 12 months from the sequential file and write a *)
 64:      (* random record after reading each monthly data set. The month *)
 65:      (* data part of the MOONDAT record is replaced by new data from *)
 66:      (* the sequential file before each write, but YEAR remains the  *)
 67:      (* same.                                                        *)
 68:
 69:  FOR MONTH:= 1 TO 12 DO
 70:   BEGIN
 71:     READLN(SRC,RF^.MONTHNAME);
 72:     WRITELN(' Moving data for ',RF^.MONTHNAME,' ',RF^.YEAR);
 73:     READLN(SRC,RF^.WD,RF^.PHASEDATE[1],RF^.PHASEDATE[2],RF^.PHASEDATE[3],
 74:            RF^.PHASEDATE[4],RF^.PHASEDATE[5]);
 75:
 76:       (* For reading in 64 array elements, a READ statement is more *)
 77:       (* convenient than READLN since you can avoid having to list  *)
 78:       (* the input target variables explicitly. But you do have to  *)
 79:       (* read a carriage return to go to the next line when it's    *)
 80:       (* time to start reading data for a new month.                *)
 81:
 82:     FOR WEEK:= 1 TO 4 DO
 83:      BEGIN
 84:        FOR K:= 1 TO 16 DO
 85:         BEGIN
 86:          READ(SRC,RF^.MONTHDATA[WEEK,K]);
 87:         END  (* FOR K *)
 88:      END;(* FOR WEEK *)
 89:     READLN(SRC);                          (* go to the next line *)
 90:     SEEKWRITE(RF,MONTH)           (* random write record number 'MONTH' *)
 91:   END; (* FOR MONTH *)
 92:   CLOSE(RF,ERR)                   (* close random file *)
 93: END; (* movedata *)
 94:
 95: BEGIN
 96:  MOVEDATATORANDOMFILE('MOONDAT','MOONRAN');
 97:  OPEN(RF,'MOONRAN',ERR);           (* open random file *)
 98:  SEEKREAD(RF,11);
 99:  WRITELN(RF^.MONTHNAME);
100:  FOR II:=1 TO 5 DO WRITELN(RF^.PHASEDATE[II]);
101:  FOR II:= 1 TO 16 DO WRITELN(RF^.MONTHDATA[1,II])
102: END.
103:
```

```
(* ************************************************************************
*
* MODULE INMOON
*
* Programmer:   Susan R. Wheler, Ga Tech/GTRI
* Source file:  INMOON.SRC
* Last revised: 8/27/84
*
* This the input module for moonrise/moonset times.
*
************************************************************************ *)

MODULE INMOON;

VAR
 ANNUM,DAY,JDATE,MONTH: EXTERNAL INTEGER;
 LATM,LONGM: EXTERNAL INTEGER;
 LAT,MINUTES,LONG: EXTERNAL REAL;
 LATD,LONGD:  EXTERNAL CHAR;
 MDAY: ARRAY [1..12] OF INTEGER;
 DEG: INTEGER;


EXTERNAL FUNCTION INTIN(PROMPT:STRING;MIN,MAX:INTEGER):INTEGER; (* BEESLIB *)
EXTERNAL PROCEDURE GET_NEXT_KEY_PRESSED(VAR C: CHAR);          (* BEESLIB *)
EXTERNAL PROCEDURE PRINTMOONHEADING;                           (* MOON    *)

PROCEDURE ASSIGN_MONTHS;

BEGIN
  MDAY[1]:= 0;                          (* store julian date for start *)
  MDAY[2]:= 31;                         (* of the 12 months in array   *)
  MDAY[3]:= 59;                         (* MDAY.                       *)
  MDAY[4]:= 90;
  MDAY[5]:= 120;
  MDAY[6]:= 151;
  MDAY[7]:= 181;
  MDAY[8]:= 212;
  MDAY[9]:= 243;
  MDAY[10]:= 273;
  MDAY[11]:= 304;
  MDAY[12]:= 334;
END;

PROCEDURE GET_DATE_AND_LOCATION;

BEGIN
  ASSIGN_MONTHS;
  PRINTMOONHEADING;
  WRITELN;WRITELN('Enter DATE for moonrise/moonset times:');
  WRITELN;
  MONTH:= INTIN('  MONTH (1-12): ',1,12);
  DAY:=   INTIN('  DAY   (1-31): ',1,31);
  IF (ANNUM/4 = ANNUM DIV 4) AND (MONTH > 2)
     THEN JDATE:= DAY + MDAY[MONTH] + 1              (* Julian date *)
     ELSE JDATE:= DAY + MDAY[MONTH];
  WRITELN;WRITELN('Enter LATITUDE of moonrise/moonset location:');
  WRITELN;
  DEG:= INTIN('  Degrees (0-60): ',0,60);
  LATM:= INTIN('  Minutes (0-60): ',0,60);
```

```
61:   MINUTES:= LATM;
62:   LAT:= DEG + MINUTES/60;
63:   WRITE ('  Direction from equator (N/S): ');
64:   REPEAT GET_NEXT_KEY_PRESSED(LATD);
65:   UNTIL LATD IN ['S','N'];
66:   WRITE(LATD);
67:   READLN;
68:   IF LATD = 'S'  THEN LAT:= -1*LAT;
69:   WRITELN;
70:   WRITELN('Enter LONGITUDE for moonrise/moonset location:');
71:   WRITELN;
72:   DEG:= INTIN('  Degrees (0-180): ',0,180);
73:   LONGM:= INTIN('  Minutes (0-60): ',0,60);
74:   WRITE ('  Direction from Greenwich (E/W): ');
75:   REPEAT GET_NEXT_KEY_PRESSED(LONGD)
76:     UNTIL LONGD IN ['E','W'];
77:   WRITE(LONGD);
78:   READLN;
79:   MINUTES:= LONGM;
80:   LONG:= DEG + MINUTES/60;
81:   IF LONGD ='W' THEN LONG:= -1 *LONG;
82: END;
83: MODEND.
84:
```

```
(* ********************************************************************************
*
* MODULE MOON
*
* Programmer:        Susan R. Wheeler, Georgia Tech/GTRI)
* Source file:       MOON.SRC
* Last modified:     8/31/84
*
* This is the main module for the Moon Phenomena program which computes
* moonrise/moonset times for geographic locations between 60 deg N and
* 60 deg  S.
*
* Accuracy inside the allowable location range should be within 5 minutes.
* This program requires an auxiliary data file to execute (see the programmer
* reference documentation for more information) which must be updated
* annually. Moonrise/moonset times are only calculated for the 'current
* year' as determined by the contents of the data file.
*
* The Moon Phenomena program can also output dates for moon phases in months
* of the current year.
*
******************************************************************************** *)


PROGRAM TMOON;


TYPE
 MOONDATA = RECORD                    (* Define record structure for random *)
          MONTHNAME: STRING[9];       (* access file containing moon data   *)
          YEAR: INTEGER;              (* for the current year.              *)
          RA: INTEGER;
          RD5: REAL;
          WD: INTEGER;
          PHASEDATE:  ARRAY[1..5] OF INTEGER;
          MONTHDATA:  ARRAY[1..4,1..16] OF REAL;;
         END;

VAR
 CHAINFIL: FILE;
 FULL,NEW,OPTION,PHENOM,WANE,WAX: STRING[14];
 DAYS,HOURS,LAT,LONG,MINUTES: REAL;
 B1,B2,B3,B4,B5,B6,B7,D5,H0,H1,T0,T1,T2,W,X,X2: REAL;
 GHA,DEC: REAL;
 DELH,GHA0,TAU,TI: REAL;
 TERM: ARRAY[1..16] OF REAL;
 A,ANNUM,CHOICE,DAY,ERR,JDATE,LAST,LATM,LONGM: INTEGER;
 MONTH,ICOUNT,RISING,SETTING,TIME,WEEK: INTEGER;
 LATD,LONGD: CHAR;
 RF: FILE OF MOONDATA;
 FILE_NOT_THERE: BOOLEAN;
 ILINE: STRING[3];


EXTERNAL FUNCTION ACOS(ARG: REAL) : REAL;                  (*  BEESLIB *)
EXTERNAL FUNCTION ASIN(ARG: REAL) : REAL;                  (*     "    *)
EXTERNAL FUNCTION CHARTOINTEGER(LIN: STRING): INTEGER;     (*     "    *)
EXTERNAL FUNCTION DTR(ARG: REAL) : REAL;                   (*     "    *)
EXTERNAL FUNCTION RMOD(ARG, MODF: REAL) : REAL;            (*     "    *)
```

```
 61: EXTERNAL FUNCTION RTD(ARG: REAL) : REAL;                    (*     "    *)
 62: EXTERNAL PROCEDURE GETMENUCHOICE(MAX:INTEGER;VAR C:INTEGER);(*   "    *)
 63: EXTERNAL PROCEDURE CLRSCR;                                  (*  CURLIB  *)
 64: EXTERNAL PROCEDURE GET_DATE_AND_LOCATION;                   (*  INMOON  *)
 65: EXTERNAL PROCEDURE GET_NEXT_KEY_PRESSED(VAR C: CHAR);       (*  BEESLIB *)
 66: EXTERNAL PROCEDURE GET_A_YESNO_ANSWER(VAR C: CHAR);         (*  BEESLIB *)
 67: EXTERNAL PROCEDURE GETLOCALTIME(LONG: REAL);                (*  BEESLIB *)
 68: EXTERNAL PROCEDURE GOTOXY(X,Y: BYTE);                       (*  CURLIB  *)
 69: EXTERNAL PROCEDURE MENU(NAME:STRING);                       (*  CURLIB  *)
 70: EXTERNAL PROCEDURE PAUSE;                                   (*  BEESLIB *)
 71:
 72:
 73: PROCEDURE CHOOSE_PROGRAM_OPTION;
 74:
 75: BEGIN
 76:  PRINT_MOON_HEADING;
 77:  MENU('MOON.MNU');
 78:  GETMENUCHOICE(2,CHOICE);
 79:  CASE CHOICE OF
 80:    0:  OPTION:= 'QUIT';
 81:    1:  OPTION:= 'RISE&SET';
 82:    2:  OPTION:= 'PHASES';
 83:   END;   (* case *)
 84: END;
 85:
 86:
 87: PROCEDURE FIND_DATA;
 88:
 89: VAR I: INTEGER;
 90:
 91: BEGIN
 92:  WEEK:= (DAY - 1) DIV 8;       (* integer divide *)
 93:  WEEK:= WEEK + 1;
 94:  SEEKREAD(RF,MONTH);           (* random read record for current month *)
 95:  FOR I:= 1 TO 16 DO TERM[I]:= RF^.MONTHDATA[WEEK,I];
 96: END;
 97:
 98:
 99: PROCEDURE GET_MONTH_AND_PRINT_PHASES;
100:
101: VAR I: INTEGER;
102:
103: BEGIN
104:  PRINTMOONHEADING;
105:  WRITELN;WRITELN;
106:  REPEAT
107:    WRITE('Which month? (1-12) ');
108:    READLN(ILINE);
109:    MONTH:= CHARTOINTEGER(ILINE)
110:   UNTIL (MONTH >= 1) AND (MONTH <= 12);
111:  SEEKREAD(RF,MONTH);       (* random read, record # = MONTH *)
112:  PRINT_MOON_HEADING;
113:  GOTOXY(10,6);
114:  WRITELN('Approximate moon phase dates for ',RF^.MONTHNAME,' ',RF^.YEAR);
115:  WRITELN('     ------------------------------------------------------');
116:  WRITELN;WRITELN;
117:  CASE RF^.WD OF
118:   1:  WRITELN(WANE,NEW,WAX,FULL);
119:   2:  WRITELN(NEW,WAX,FULL,WANE);
120:   3:  WRITELN(WAX,FULL,WANE,NEW);
```

```
121:   4:  WRITELN(FULL,WANE,NEW,WAX);
122:   10: WRITELN(WANE,NEW,WAX,FULL,WANE);
123:   20: WRITELN(NEW,WAX,FULL,WANE,NEW);
124:   30: WRITELN(WAX,FULL,WANE,NEW,WAX);
125:   40: WRITELN(FULL,WANE,NEW,WAX,FULL)
126:   END;
127:  IF RF^.WD < 10 THEN LAST:= 4 ELSE LAST:= 5;
128:  WRITELN;
129:  FOR I:= 1 TO LAST DO WRITE(RF^.PHASEDATE[I]:7,'       ');
130:  PAUSE
131: END;
132:
133: PROCEDURE GHA_AND_DEC;          (* compute moon GHA and dec for *)
134:                                 (* current iteration            *)
135: BEGIN
136:  W:= TRUNC((DAY - 1) / 8);
137:  W:= W * 8 + 1;
138:    (* series expansion *)
139:  X:= (DAY + T1 - W ) / A - 1;
140:  B7:= X * TERM[8];
141:  B6:= X * (TERM[7] + B7);
142:  B5:= X * (TERM[6] + B6);
143:  B4:= X * (TERM[5] + B5);
144:  B3:= X * (TERM[4] + B4);
145:  B2:= X * (TERM[3] + B3);
146:  B1:= X * (TERM[2] + B2);
147:  GHA:= RMOD((TERM[1] + B1),360);  (* moon Greenwich Hour Angle at time T0 *)
148:  B7:= X * TERM[16];
149:  B6:= X * (TERM[15] + B7);
150:  B5:= X * (TERM[14] + B6);
151:  B4:= X * (TERM[13] + B5);
152:  B3:= X * (TERM[12] + B4);
153:  B2:= X * (TERM[11] + B3);
154:  B1:= X * (TERM[10] + B2);
155:  DEC:= RMOD((TERM[9] + B1),360);  (* moon declination at time T0 *)
156: END;
157:
158: PROCEDURE ITERATE;
159:
160: BEGIN
161:  X2:= (0.00233-SIN(DTR(LAT)) * SIN(DTR(DEC)))/ (COS(DTR(LAT)) * COS(DTR(DEC))
162: );
163:  H1:= RTD ( ACOS(X2));    (* H1 = approx to moon local hour angle *)
164:  IF TIME = RISING THEN H1:= 360 - H1;
165:  TAU:= (H1 - H0) / DELH;  (* TAU = correction to 1st guess for phenom time *)
166:  IF TAU > 0.5 THEN TAU:= TAU - 360 / DELH
167:  ELSE
168:   IF  TAU < -0.5 THEN TAU:= TAU + 360 / DELH;
169:  T2:= T0 + TAU;
170:  IF ABS(T2-T1) < 0.01 THEN PRINTOUTPUT  (* computed time is good enough *)
171:  ELSE BEGIN
172:   ICOUNT:= ICOUNT + 1;
173:   IF ICOUNT > 12 THEN PRINT_OUTPUT      (* do no more than 12 iterations *)
174:   ELSE BEGIN
175:     T1:=T2;
176:     GHAANDDEC;
177:     DELH:= (GHA - GHA0) / TAU;   (* compute delh for iterations past the 1st
178: *)
179:     IF DELH < 0 THEN DELH:= DELH + 360 / ABS(TAU);
180:     ITERATE
```

```
181:    END
182:  END
183: END;
184:
185:
186: PROCEDURE PHENOMENA_TIME_CALCULATIONS;
187:
188: BEGIN
189:                (* D5 = approximation to moon's daily rate of change in *)
190:                (* GHA. The value to use on iteration 0 is a constant   *)
191:                (* which is updated annually.                           *)
192:
193:  SEEKREAD(RF,1);
194:  D5:= RF^.RD5;   (* Get D5 off of record 1 in the moon data file.       *)
195:  ICOUNT:= 0;
196:  T0:= (12 - LONG / 15 ) / 24;  (* initial approximation to phenomenon time *)
197:  T1:= T0;
198:  GHAANDDEC;
199:  DELH:= D5;           (* initial approx daily rate of change in GHA *)
200:  GHA0:= GHA;          (* save GHA from iteration 0 *)
201:  H0:= GHA0 + LONG;    (* H0 = moon local hour angle on iteration 0 *)
202:  ITERATE;
203: END;
204:
205: PROCEDURE PRINTINPUTS;
206:
207: BEGIN
208:
209:   PRINTMOONHEADING;
210:   WRITELN;
211:   WRITE('                ');
212:   WRITELN('Date:         ',MONTH:2,'/',TRUNC(DAY):2:0,'/',RF^.YEAR:4);
213:   WRITE('                ');
214:   WRITELN('Latitude:   ',ABS(TRUNC(LAT)):4:0,' Degrees',LATM:3,
215:           ' Minutes ',LATD);
216:   WRITE('                ');
217:   WRITELN('Longitude: ',ABS(TRUNC(LONG)):4:0,' Degrees',LONGM:3,
218:           ' Minutes ',LONGD);
219:   WRITELN;
220:   GETLOCALTIME(LONG);
221:   WRITELN
222: END;
223:
224:
225: PROCEDURE PRINT_MOON_HEADING;      (* clear screen and print a heading *)
226:                                    (* for this program                 *)
227: BEGIN
228:  CLRSCR;
229:  WRITELN('                  MICROFIX BEES:    MOON PHENOMENA');
230:  WRITELN('-------------------------------------------------------------',
231:          '--------');
232:  WRITELN('-------------------------------------------------------------',
233:          '--------')
234: END;
235:
236:
237: PROCEDURE PRINT_OUTPUT;
238:
239: BEGIN
240:  DAYS:= T2 + JDATE;
```

```
241:  HOURS:= (DAYS - TRUNC(DAYS)) * 24;
242:  MINUTES:= (HOURS - TRUNC(HOURS)) * 60;
243:  IF TIME = RISING THEN BEGIN
244:    PHENOM:= 'MOONRISE';
245:    WRITELN('                        MOON RISE AND SET TIMES');
246:    WRITELN('                        ----------------------');
247:    WRITELN;
248:    WRITELN('                              JULIAN DATE        ZULU TIME');
249:    WRITELN
250:   END
251:  ELSE PHENOM:= 'MOONSET ';
252:  IF ICOUNT < 13 THEN WRITELN('          ',PHENOM,TRUNC(DAYS):15:0,
253:           TRUNC(HOURS):15:0,':',TRUNC(MINUTES):3:0)
254:  ELSE WRITELN('          ',PHENOM,'   None expected. Check preceding ',
255:               'and following dates.');
256:  IF TIME = SETTING THEN BEGIN
257:    WRITELN;
258:    WRITELN('          Values are accurate to + or - 5 minutes.');
259:    PAUSE
260:   END
261: END;
262:
263:
264: PROCEDURE SETUP;
265:
266: BEGIN
267:  OPEN(RF,'MOONRAN',ERR);       (* Open random moon data file *)
268:  IF ERR = 255 THEN FILE_NOT_THERE:= TRUE
269:   ELSE BEGIN
270:         FILE_NOT_THERE:= FALSE;
271:         SEEKREAD(RF,1);
272:         ANNUM:= RF^.YEAR       (* get year from the 1st record *)
273:        END;
274:  SEEKREAD(RF,1);
275:  A:=RF^.RA;                   (* annual updated value from file *)
276:  WAX:=  'CRESCENT/WAX   ';
277:  WANE:= 'CRESCENT/WANE  ';
278:  FULL:= 'FULL MOON      ';
279:  NEW:=  'NEW MOON       ';
280:  RISING:= 1;
281:  SETTING:= 2
282: END;
283:
284: BEGIN
285:  SETUP;
286:  IF FILE_NOT_THERE THEN BEGIN
287:     WRITELN('Data file is missing.');
288:     WRITELN('Sorry, program must be terminated.')
289:   END
290:  ELSE REPEAT
291:    CHOOSE_PROGRAM_OPTION;
292:    IF  OPTION='PHASES' THEN GET_MONTH_AND_PRINT_PHASE_DATES
293:    ELSE IF OPTION = 'RISE&SET' THEN BEGIN
294:      GET_DATE_AND_LOCATION;
295:      PRINTINPUTS;
296:      FIND_DATA;
297:      FOR TIME:= RISING TO SETTING DO PHENOMENA_TIME_CALCULATIONS;
298:     END
```

```
301:  RESET(CHAINFIL);
302:  CHAIN(CHAINFIL)                    (* chain back to driver *)
303:
304: END.
305:
```

```
(* ************************************************************************
*
*  MODULE INSUN
*
* Programmer:    Susan R. Wheeler, Ga Tech/EES
* Source file:   INSUN.SRC
* Last revised: 8/27/84
*
* This is the input module for the sun phenomena computation program.
* It solicits date and latitude/longitude/direction information from
* the user. The input information is available to the main module, as
* all variables indicated EXTERNAL to this module are declared in the
* main module SUNRISE.
*
************************************************************************ *)

MODULE INSUN;

VAR
 LATM,LONGM,MONTH,DAY,YEAR,JDATE: EXTERNAL INTEGER;
 LAT,MINUTES,LONG: EXTERNAL REAL;
 LATD,LONGD:  EXTERNAL CHAR;
 MDAY: ARRAY [1..12] OF INTEGER;
 DEG: INTEGER;

EXTERNAL FUNCTION INTIN(PROMPT:STRING;MAX,MIN:INTEGER):INTEGER; (* BEESLIB *)
EXTERNAL PROCEDURE GET_NEXT_KEY_PRESSED(VAR C: CHAR);          (* BEESLIB *)

PROCEDURE ASSIGN_MONTHS;

BEGIN
  MDAY[1]:= 0;                          (* Store Julian date for the beginning *)
  MDAY[2]:= 31;                         (* of the 12 months in array MDAY      *)
  MDAY[3]:= 59;
  MDAY[4]:= 90;
  MDAY[5]:= 120;
  MDAY[6]:= 151;
  MDAY[7]:= 181;
  MDAY[8]:= 212;
  MDAY[9]:= 243;
  MDAY[10]:= 273;
  MDAY[11]:= 304;
  MDAY[12]:= 334;
END;

PROCEDURE GET_INPUTS;

BEGIN
  ASSIGN_MONTHS;
  WRITELN;WRITELN('Enter DATE for sunrise/sunset times:');
  WRITELN;
  MONTH:= INTIN('  MONTH    (1-12): ',1,12);
  DAY:=   INTIN('  DAY      (1-31): ',1,31);
  YEAR:=  INTIN('  YEAR (ex: 1983): ',0,9999);
  IF (YEAR/4 = YEAR DIV 4) AND (MONTH > 2)
     THEN JDATE:= DAY + MDAY[MONTH] + 1        (* Julian date - leapyear *)
     ELSE JDATE:= DAY + MDAY[MONTH];           (*  or not leapyear *)
  WRITELN;WRITELN('Enter LATITUDE of sunrise/sunset location:');
  WRITELN;
  DEG:= INTIN('  Degrees (0-60): ',0,60);
```

```
61:   LATM:= INTIN('  Minutes (0-60): ',0,60);
62:   MINUTES:= LATM;
63:   LAT:= DEG + MINUTES/60;
64:   WRITE ('  Direction from equator (N/S): ');
65:   REPEAT GET_NEXT_KEY_PRESSED(LATD);          (* Accept n,N,s, or S answer *)
66:   UNTIL LATD IN ['S','N'];                    (* only. GETKEY converts lower *)
67:   WRITE(LATD);                                (* case input to upper case.   *)
68:   READLN;
69:   IF LATD = 'S'  THEN LAT:= -1*LAT;
70:   WRITELN;
71:   WRITELN('Enter LONGITUDE for sunrise/sunset location:');
72:   WRITELN;
73:   DEG:= INTIN('  Degrees (0-180): ',0,180);
74:   LONGM:= INTIN('  Minutes  (0-60): ',0,60);
75:   WRITE ('  Direction from Greenwich (E/W): ');
76:   REPEAT GET_NEXT_KEY_PRESSED(LONGD)             (* Get a E or W as above *)
77:     UNTIL LONGD IN ['E','W'];
78:   WRITE(LONGD);
79:   READLN;
80:   MINUTES:= LONGM;
81:   LONG:= DEG + MINUTES / 60;
82:   IF LONGD ='E' THEN LONG:= -1 *LONG;
83: END;
84: MODEND.
85:
```

```
(* ************************************************************************
*
* PROGRAM SUNRISE
*
* Programmer:   Susan R. Wheeler, Ga Tech/EES
* Source file:  SUNRISE.SRC
* Last revised: 8/23/84
*
* This is the main module for the program to calculate sunrise and sunset
* data for a user-input date and geographic location (latitude/longitude
* coordinates between 60 deg N and 60 deg S). Output is ZULU time of 4 sun
* phenomena for the specified date and location.
*
* Computation variable names and comments from the original BASIC program
* have generally been preserved. MT+ PASCAL uses radian input and output for
* trigonometric functions and has no implicit functions for ARCCOS, ARCSIN,
* and modulus of real numbers. These functions have been provided for by the
* programmer. Location of procedures and functions external to this module
* is indicated. Source code for module X may be found in file X.SRC.
*
* The sun program is chained to, and chains back to the main BEES menu.
*
************************************************************************ *)

PROGRAM SUNRISE;

VAR

  MONTH,DAY,YEAR,JDATE: INTEGER;
  LAT,MINUTES,LONG: REAL;
  LATM,LONGM: INTEGER;
  LATD,LONGD,ANSWER: CHAR;
  T,MA,L,RA,D,Z,H,T1,T2,SPAN: REAL;
  S1,S2,H1,H2,M1,M2: INTEGER;
  W: STRING;
  CHAINFIL: FILE;

EXTERNAL FUNCTION ACOS(ARG: REAL) : REAL;              (*  BEESLIB *)
EXTERNAL FUNCTION ASIN(ARG: REAL) : REAL;              (*     "    *)
EXTERNAL FUNCTION DTR(ARG: REAL) : REAL;               (*     "    *)
EXTERNAL FUNCTION RMOD(ARG, MODF: REAL) : REAL;        (*     "    *)
EXTERNAL FUNCTION RTD(ARG: REAL) : REAL;               (*     "    *)
EXTERNAL PROCEDURE CLRSCR;                             (*  CURLIB  *)
EXTERNAL PROCEDURE GETLOCALTIME(LONG: REAL);           (*  BEESLIB *)
EXTERNAL PROCEDURE GET_INPUTS;                         (*  INSUN   *)
EXTERNAL PROCEDURE GET_A_YESNO_ANSWER(VAR C: CHAR);    (*  BEESLIB *)

PROCEDURE COMPUTE_PHENOMENA_TIMES;          {* computation loop for four *}
                                            {*      sun phenomena        *}
VAR
 I: INTEGER;

BEGIN
  FOR I:= 1 TO 4 DO
   BEGIN
     IF I = 1 THEN  BEGIN
         Z:= -0.0145439;
         W:= 'Sunrise and Sunset
       END;
     IF I = 2 THEN  BEGIN
```

```
          Z:= -0.1045285;
          W:= 'Civil Twilight          '
        END;
      IF I = 3 THEN  BEGIN
          Z:= -0.2079117;
          W:= 'Nautical Twilight       '
        END;
      IF I = 4 THEN  BEGIN
          Z:= -0.309017;
          W:= 'Astronomical Twilight   '
        END;

     {* Compute sun's local hour angle. D is already in radians *}
     (* Latitude and longitude must be changed to radians from degrees *)

    H:= (Z - SIN(D) * SIN(DTR(LAT))) /(COS(D) * COS(DTR(LAT)));
    IF ABS(H) (= 1 THEN
      BEGIN
        H:= ACOS(H);   {* arccos h (radian output-change to degrees below) *}
        T1:= (360 - RTD(H)) / 15 + RA - 0.06571 * T - 6.622 + LONG / 15;
        T1:= RMOD(T1,24);   {* rising phenom's local time = T1 MOD 24 *}
        T2:= (RTD(H)  / 15 + RA - 0.06571 * T - 6.622 + LONG / 15);
        T2:= RMOD(T2,24);   {* setting phenom local time *}
        IF T1 ( T2 THEN SPAN:= ABS (T1 - T2)
         ELSE SPAN:= ABS (T2 + 24 - T1);   {* time span computed *}
        S1:= TRUNC(SPAN);
        S2:= TRUNC((SPAN - TRUNC(SPAN)) * 60);
        H1:= TRUNC(T1);
        H2:= TRUNC(T2);
        M1:= TRUNC((T1 - TRUNC(T1)) * 60);
        M2:= TRUNC((T2 - TRUNC(T2)) * 60);


                    (* printout for time of phenomena *)

        WRITELN(W,H1:2,':',M1:2,' ZULU  ',H2:2,':',M2:2,' ZULU   ',
                S1:2,' HOURS ',S2:2,' MINUTES');
      END  {* IF *}
     ELSE          {* printout for absence of phenomenon (/cos(h)/ ) 0 ) *}
        WRITELN(W,'       Twilight lasts all night.');
     END;    {* FOR *}
    WRITELN;WRITELN(' Values are accurate to within + or - 5 minutes.');
    WRITELN
END;  {* COMPUTE *}


PROCEDURE PRINT_SUN_HEADING;      (* clear screen and print a heading *)
                                  (* for this program                 *)
BEGIN
 CLRSCR;
 WRITELN('      MICROFIX BEES:      SUNRISE/SUNSET/TWILIGHT TIMES');
 WRITELN('                        (Valid for latitudes 60degN to 60degS)');
 WRITELN('-------------------------------------------------------------',
         '--------');
 WRITELN('-------------------------------------------------------------',
         '--------')
END;


PROCEDURE Find_right_ascension_and_declination;
```

```
121:
122:  BEGIN
123:   T:= JDATE + (6 + LONG/15) / 24; {* approx days since 0 Jan, 0 hrs UT *}
124:   MA:= 0.9856 * T - 3.289;        {* sun's mean anomoly *}
125:
126:         {* Compute sun's true longitude *}
127:         (* Arguments of sin and cos must be converted to radians *)
128:
129:   L:= (MA + 1.916 * SIN(DTR(MA)) + 0.02 * SIN(DTR(2*MA)) + 282.634);
130:   L:= RMOD(L,360);                    {* MOD 360 *}
131:   IF L = 180 THEN L:= 179.999999
132:     ELSE IF L = 270 THEN L:= 269.999999;
133:
134:         {* Compute sun's right ascension (RA) *}
135:
136:   RA:= ARCTAN(0.91746 * SIN(DTR(L))/COS(DTR(L)));
137:   RA:= RTD (RA);     (* Convert radian output from ARCTAN to degrees *)
138:
139:          {* Place RA in same quadrant as L *}
140:
141:   IF (L ) 90) AND (L ( 180) THEN RA:= 90 - ABS(RA) + 90;
142:   IF (L )= 180) AND (L ( 270) THEN RA:= ABS(RA) + 180;
143:   IF L ) 270 THEN RA:= 90 - ABS(RA) + 270;
144:   RA:= RA/15;              {* Convert hours to degrees *}
145:   D:= 0.39782 * SIN(DTR(L));
146:   D:= ASIN(D);  {* Sun's declination = arcsin(D) *}
147:  END;     (* Find RA and D *)
148:
149:
150: PROCEDURE GET_DATE_AND_LOCATION;       (* Get inputs from user and print    *)
151:                                         (* preliminary output. The times of  *)
152:  BEGIN                                  (* phenomena are printed as they are *)
153:   PRINT_SUN_HEADING;                    (* computed.                         *)
154:   GET_INPUTS;
155:   PRINT_SUN_HEADING;
156:   WRITELN;
157:   WRITE('                ');
158:   WRITELN('Date:         ',MONTH:2,'/',TRUNC(DAY):2:0,'/',YEAR:4);
159:   WRITE('                ');
160:   WRITELN('Latitude:  ',ABS(TRUNC(LAT)):4:0,' Degrees',LATM:3,
161:           ' Minutes ',LATD);
162:   WRITE('                ');
163:   WRITELN('Longitude: ',ABS(TRUNC(LONG)):4:0,' Degrees',LONGM:3,
164:           ' Minutes ',LONGD);
165:   WRITELN;
166:   GETLOCALTIME(LONG);
167:   WRITELN;
168:   WRITE('                              ');
169:   WRITELN('RISING      SETTING          DURATION');
170:   WRITE('                              ');
171:   WRITELN('------      -------          --------')
172:  END;
173:
174:
175:              (* MAIN PROCEDURE *)
176:
177: BEGIN
178:   REPEAT
179:     GET_DATE_AND_LOCATION;
180:     FIND_RIGHT_ASCENSION_AND_DECLINATION;
```

```
181:     COMPUTE_PHENOMENA_TIMES;
182:     WRITE('Do you want sunrise/sunset for another location? (Y/N) ');
183:     GET_A_YESNO_ANSWER(ANSWER)
184:   UNTIL ANSWER = 'N';
185:   ASSIGN(CHAINFIL,'BEES.COM');     (* chain back to main menu *)
186:   RESET(CHAINFIL);
187:   CHAIN(CHAINFIL)
188: END.
189:
```

Section 4

UTILITIES

(THIS SECTION IS INTENTIONALLY BLANK)

Section 5

TECHNICAL ITEMS

Almanac for Computers

1984

Nautical Almanac Office
United States Naval Observatory
Washington, D. C. 20390

Table of Contents

Section A: EXPLANATION

Introduction (with material that should be read)

Incorporated in *A/C 84* are a number of changes from previous editions. Most obvious is the use of the day of the month rather than day of the year as the time argument in the navigation tables (Section C). To facilitate this feature the power series for lunar coordinates now cover spans of eight rather than six days. The price to be paid is that eight rather than six terms are required for the power series. In Section B formulas involving geographic longitudes have been revised so that longitudes east of Greenwich are positive and west longitudes are negative. This conforms to the convention that has been common in many fields and has been newly adopted in astronomy. In Section E apparent places of the stars are now computed from mean places for the middle rather than the beginning of the year. But since this change is accounted for in adjusted values of the constants, no reprogramming should be necessary.

The almanacs for 1984 are based on a new systen of planetary ephemerides, precession and nutation expressions, time scale (dynamical time replacing ephemeris time), standard epoch (J2000.0 replacing 1950.0 and 1900.0), equinox (FK5 replacing FK4), and definition of astrometric position. This new system forms the basis for *A/C 84*. Information on the new system is given in *The Astronomical Almanac 1984*, the *Supplement to the Astronomical Almanac 1984*, and *U. S. Naval Observatory Circular 163*.

For most efficient use with computers, the data in Sections D and E are available on magnetic tape (data in Section C are not available in machine readable form). Inquiries about this service, as well as comments and suggestions concerning this volume, should be addressed to The Director, Nautical Almanac Office, U. S. Naval Observatory, Washington, D. C. 20390.

This volume was produced by LeRoy E. Doggett and Patrick E. Protacio, Ensign, USN.

CALENDAR, 1984

	JANUARY		FEBRUARY		MARCH		APRIL		MAY		JUNE	
Day of Month	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year
1	Sun.	1	Wed.	32	Thu.	61	Sun.	92	Tue.	122	Fri.	153
2	Mon.	2	Thu.	33	Fri.	62	Mon.	93	Wed.	123	Sat.	154
3	Tue.	3	Fri.	34	Sat.	63	Tue.	94	Thu.	124	Sun.	155
4	Wed.	4	Sat.	35	Sun.	64	Wed.	95	Fri.	125	Mon.	156
5	Thu.	5	Sun.	36	Mon.	65	Thu.	96	Sat.	126	Tue.	157
6	Fri.	6	Mon.	37	Tue.	66	Fri.	97	Sun.	127	Wed.	158
7	Sat.	7	Tue.	38	Wed.	67	Sat.	98	Mon.	128	Thu.	159
8	Sun.	8	Wed.	39	Thu.	68	Sun.	99	Tue.	129	Fri.	160
9	Mon.	9	Thu.	40	Fri.	69	Mon.	100	Wed.	130	Sat.	161
10	Tue.	10	Fri.	41	Sat.	70	Tue.	101	Thu.	131	Sun.	162
11	Wed.	11	Sat.	42	Sun.	71	Wed.	102	Fri.	132	Mon.	163
12	Thu.	12	Sun.	43	Mon.	72	Thu.	103	Sat.	133	Tue.	164
13	Fri.	13	Mon.	44	Tue.	73	Fri.	104	Sun.	134	Wed.	165
14	Sat.	14	Tue.	45	Wed.	74	Sat.	105	Mon.	135	Thu.	166
15	Sun.	15	Wed.	46	Thu.	75	Sun.	106	Tue.	136	Fri.	167
16	Mon.	16	Thu.	47	Fri.	76	Mon.	107	Wed.	137	Sat.	168
17	Tue.	17	Fri.	48	Sat.	77	Tue.	108	Thu.	138	Sun.	169
18	Wed.	18	Sat.	49	Sun.	78	Wed.	109	Fri.	139	Mon.	170
19	Thu.	19	Sun.	50	Mon.	79	Thu.	110	Sat.	140	Tue.	171
20	Fri.	20	Mon.	51	Tue.	80	Fri.	111	Sun.	141	Wed.	172
21	Sat.	21	Tue.	52	Wed.	81	Sat.	112	Mon.	142	Thu.	173
22	Sun.	22	Wed.	53	Thu.	82	Sun.	113	Tue.	143	Fri.	174
23	Mon.	23	Thu.	54	Fri.	83	Mon.	114	Wed.	144	Sat.	175
24	Tue.	24	Fri.	55	Sat.	84	Tue.	115	Thu.	145	Sun.	176
25	Wed.	25	Sat.	56	Sun.	85	Wed.	116	Fri.	146	Mon.	177
26	Thu.	26	Sun.	57	Mon.	86	Thu.	117	Sat.	147	Tue.	178
27	Fri.	27	Mon.	58	Tue.	87	Fri.	118	Sun.	148	Wed.	179
28	Sat.	28	Tue.	59	Wed.	88	Sat.	119	Mon.	149	Thu.	180
29	Sun.	29	Wed.	60	Thu.	89	Sun.	120	Tue.	150	Fri.	181
30	Mon.	30			Fri.	90	Mon.	121	Wed.	151	Sat.	182
31	Tue.	31			Sat.	91			Thu.	152		

JULIAN DATE, 1984

0^h UT				0^h UT				0^h UT				0^h UT			
Jan.	0	244	5699·5	Apr.	0	244	5790·5	July	0	244	5881·5	Oct.	0	244	5973·5
Feb.	0	244	5730·5	May	0	244	5820·5	Aug.	0	244	5912·5	Nov.	0	244	6004·5
Mar.	0	244	5759·5	June	0	244	5851·5	Sept.	0	244	5943·5	Dec.	0	244	6034·5

400-day date: JD 244 6000·5 = 1984 October 27·0

CALENDAR, 1984

	JULY		AUGUST		SEPTEMBER		OCTOBER		NOVEMBER		DECEMBER	
Day of Month	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year	Day of Week	Day of Year
1	Sun.	183	Wed.	214	Sat.	245	Mon.	275	Thu.	306	Sat.	336
2	Mon.	184	Thu.	215	Sun.	246	Tue.	276	Fri.	307	Sun.	337
3	Tue.	185	Fri.	216	Mon.	247	Wed.	277	Sat.	308	Mon.	338
4	Wed.	186	Sat.	217	Tue.	248	Thu.	278	Sun.	309	Tue.	339
5	Thu.	187	Sun.	218	Wed.	249	Fri.	279	Mon.	310	Wed.	340
6	Fri.	188	Mon.	219	Thu.	250	Sat.	280	Tue.	311	Thu.	341
7	Sat.	189	Tue.	220	Fri.	251	Sun.	281	Wed.	312	Fri.	342
8	Sun.	190	Wed.	221	Sat.	252	Mon.	282	Thu.	313	Sat.	343
9	Mon.	191	Thu.	222	Sun.	253	Tue.	283	Fri.	314	Sun.	344
10	Tue.	192	Fri.	223	Mon.	254	Wed.	284	Sat.	315	Mon.	345
11	Wed.	193	Sat.	224	Tue.	255	Thu.	285	Sun.	316	Tue.	346
12	Thu.	194	Sun.	225	Wed.	256	Fri.	286	Mon.	317	Wed.	347
13	Fri.	195	Mon.	226	Thu.	257	Sat.	287	Tue.	318	Thu.	348
14	Sat.	196	Tue.	227	Fri.	258	Sun.	288	Wed.	319	Fri.	349
15	Sun.	197	Wed.	228	Sat.	259	Mon.	289	Thu.	320	Sat.	350
16	Mon.	198	Thu.	229	Sun.	260	Tue.	290	Fri.	321	Sun	351
17	Tue.	199	Fri.	230	Mon.	261	Wed.	291	Sat.	322	Mon.	352
18	Wed.	200	Sat.	231	Tue.	262	Thu.	292	Sun.	323	Tue.	353
19	Thu.	201	Sun.	232	Wed.	263	Fri.	293	Mon.	324	Wed.	354
20	Fri.	202	Mon.	233	Thu.	264	Sat.	294	Tue.	325	Thu.	355
21	Sat.	203	Tue.	234	Fri.	265	Sun.	295	Wed.	326	Fri.	356
22	Sun.	204	Wed.	235	Sat.	266	Mon.	296	Thu.	327	Sat.	357
23	Mon.	205	Thu.	236	Sun.	267	Tue.	297	Fri.	328	Sun.	358
24	Tue.	206	Fri.	237	Mon.	268	Wed.	298	Sat.	329	Mon.	359
25	Wed.	207	Sat.	238	Tue.	269	Thu.	299	Sun.	330	Tue.	360
26	Thu.	208	Sun.	239	Wed.	270	Fri.	300	Mon.	331	Wed.	361
27	Fri.	209	Mon.	240	Thu.	271	Sat.	301	Tue.	332	Thu.	362
28	Sat.	210	Tue.	241	Fri.	272	Sun.	302	Wed	333	Fri.	363
29	Sun.	211	Wed.	242	Sat.	273	Mon.	303	Thu.	334	Sat.	364
30	Mon.	212	Thu.	243	Sun.	274	Tue.	304	Fri.	335	Sun.	365
31	Tue.	213	Fri.	244			Wed.	305			Mon.	366

MEAN SIDEREAL TIME, 1984

Greenwich mean sidereal time at 0^h UT

	h		h		h		h
Jan. 0	06·5906	Apr. 0	12·5702	July 0	18·5498	Oct. 0	00·5951
Feb. 0	08·6276	May 0	14·5415	Aug. 0	20·5868	Nov. 0	02·6321
Mar. 0	10·5332	June 0	16·5785	Sept. 0	22·6238	Dec. 0	04·6034

Navigational Tables

Section C contains mathematical representations of the following functions that are tabulated in the *Nautical Almanac* (*NA*): the GHA of Aries, the GHA and declination of the Sun, Moon and navigational planets, the semidiameter of the Sun and Moon, and the horizontal parallax of the Moon. Except in the case of the Moon, these functions are expressed for a specified time span by a power series of the form

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$$

For the Moon there are two additional terms, a_6x^6 and a_7x^7, that must be added to the series. In the series x is a time-like variable that takes on values between -1 and $+1$ over the specified time span; a_0, a_1, a_2, etc., are coefficients that are tabulated in Section C for the specified time span; and $f(x)$ represents the value of the function (e.g., the GHA of Aries) evaluated at time x.

To evaluate the series for one of the navigational functions, one must first find the set of coefficients in Section C that is applicable for the desired date. Constants A and W are given for the purpose of converting the calendar date and GMT to the time-like variable x. First compute t, the GMT measured in days and fractions thereof from the beginning of the month: $t = d + \text{GMT}/24$, where d is the day of the month at Greenwich and GMT is the Greenwich Mean Time expressed in hours. A calendar is provided on pages A2–A3. Once t has been determined, x can be computed from the relation $x = (t - W)/A - 1$. If computed correctly, the value of x will fall in the range $-1 \leqslant x \leqslant +1$.

Example 1: Compute x for later use in computing the position of the Moon on 24 July at $22^h\,30^m\,16^s$ GMT ($=0^d.9376852$).

$t = 24 + 0.9376852$. Constants for this date are found on page C14: $A = 4$ and $W = 17$. Therefore

$x = (24.9376852 - 17)/4 - 1 = +0.9844213$.

Once the variable x has been computed and the coefficients a_i have been found, the series for the desired function can be evaluated. The series can be evaluated most efficiently by computing a set of auxiliary variables, b_1, b_2, b_3, b_4, b_5 (with additional variables b_6, b_7 for the Moon), in the following order:

For the Moon: $b_7 = xa_7$; $b_6 = x(a_6 + b_7)$; $b_5 = x(a_5 + b_6)$

For the Sun, Aries, and planets: $b_5 = xa_5$

Then for all objects:

$$b_4 = x(a_4 + b_5)$$
$$b_3 = x(a_3 + b_4)$$
$$b_2 = x(a_2 + b_3)$$
$$b_1 = x(a_1 + b_2)$$
$$f(x) = a_0 + b_1$$

By using this algorithm, the series is evaluated in its nested form. For the Sun, Aries and the planets, which have six coefficients per series, the nested series may be written in the form

$$f(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + x(a_4 + xa_5)))).$$

Example 2: Compute the declination of the Moon at $22^h\,30^m\,16^s$ GMT on 24 July 1984.

From the previous example x = +0.9844213. The coefficients for the Moon's declination are found on page C14.

b_7	= .9844213 (+0.0021)	= +0.0021
b_6	= .9844213 (+0.0183 + 0.0021)	= +0.0201
b_5	= .9844213 (−0.1435 + 0.0201)	= −0.1215
b_4	= .9844213 (−0.3371 − 0.1215)	= −0.4515
b_3	= .9844213 (−1.8210 − 0.4515)	= −2.2371
b_2	= .9844213 (−1.5186 − 2.2371)	= −3.6972
b_1	= .9844213 (+20.3310 − 3.6972)	= +16.3747
f(+0.9844213) = 7.2656 + 16.3747		= +23.6403

Therefore declination = +23°38′.4

Example 3: Compute the Sun's GHA at $15^h\,12^m\,10^s$ GMT on 31 August 1984.

The constants A and W and the series coefficients are found on page C4.

$x = (t - W)/A - 1 = (31.6334491 - 1)/16 - 1 = +0.9145906$

b_5	= .9145906 (+0.0016)	= +0.0015
b_4	= .9145906 (−0.0035 + 0.0015)	= −0.0018
b_3	= .9145906 (−0.0334 − 0.0018)	= −0.0322
b_2	= .9145906 (+0.2680 − 0.0322)	= +0.2157
b_1	= .9145906 (+5760.8525 + 0.2157)	= +5269.0188
f(+0.9145906) = 5938.9845 + 5269.0188		= +11208.0033

GHA = 48°.0033 = 48°00′.2

Note that when computing the GHA, it may be necessary to reduce the final result to the range 0°–360° by subtracting multiples of 360°.

Although the series are designed to provide precision comparable to that published in the *NA*, there will be small discrepencies between the tabulated values and the values computed from the series. In such cases it should be understood that the *NA* represents the standard. Table 1 lists the largest discrepencies found from evaluating and comparing the series with the data in the *NA*.

Under no circumstances should the series be used to extrapolate data beyond the specified time intervals. Such extrapolation will lead to erroneous and useless results.

In accordance with standard practice for navigational almanacs, the time argument used in this almanac is Greenwich Mean Time (GMT), or more specifically UT1. To obtain full precision in the determined positions, the radio time signals in UTC must be corrected to UT1, or GMT, according to standard procedures. (See the paper by R.L. Duncombe and P.K. Seidelmann, 'The New UTC Time Signals', *Navigation*, 24, 160--165, 1977.)

Beneath each set of coefficients in Section C is printed the sum of the coefficients. As a check on whether the coefficients have been entered accurately into the calculator, it is recommended that the coefficients be summed and that the resulting sum be compared with the printed sum.

Table 1: Comparison of *Almanac for Computers* with *NA*

Function	No. of Terms	Span of Validity	Maximum Error
GHA of Aries	6	32 days	0'.2
Sun: GHA	"	"	0.1
Declination	"	"	0.1
Semidiameter	"	"	0.1
Moon: GHA	8	8 days	0.2
Declination	"	"	0.1
Horizontal Parallax	"	"	0.1
Semidiameter	"	"	0.1
Navigational Planets: GHA	6	32 days	0.1
Declination	"	"	0.1

A8

Astronomical Tables

Section D contains mathematical representations of data published in the *Astronomical Almanac* (A^2). Chebyshev expansions have been chosen as the means of representation since they provide efficient and accurate expressions that can be easily evaluated with a small computer. The coefficients a_i of the Chebyshev expansion

$$f(x) = a_0/2 + \sum_{i=1}^{n} a_i T_i(x)$$

are tabulated for prescribed time spans, where $f(x)$ is the function being represented, $T_i(x)$ is the Chebyshev polynomial of the first kind of the *i-th* degree, and x is the normalized time variable. Although Chebyshev polynomials appear in the series expansions, the series can be evaluated without explicitly computing these polynomials. No *a prori* knowledge of Chebyshev analysis is required to use the series in this almanac. Interested readers can find information on Chebyshev analysis in *Applied Analysis* by C. Lanczos and *Chebyshev Polynomials in Numerical Analysis* by L. Fox and I. B. Parker.

It must be emphasized that the series are valid only over the specified time intervals. Attempts to extrapolate data using these series will yield erroneous and useless results.

If precision comparable to that of the A^2 is required, the series on pages D2-D29 should be used. With the exception of the series for the Moon, these series are valid for time spans of approximately three months; for the Moon the span of validity is approximately one month. Table 2 lists the largest errors found by evaluating these series and comparing the results with data printed in the A^2.

It is possible to develop series that are valid for longer time spans if the precision requirements are relaxed. Such series, valid for one full year, are given on pages D30–D33. Precision criteria of these less precise series are summarized in Table 3.

To evaluate a Chebyshev series, one must first normalize the time variable on the interval for which the series is valid. The normalized time x can be determined from the relation $x = (t - W)/A - 1$, where t is reckoned in days and fractions thereof from 0 January. As in previous editions, constants A and W are given for each set of coefficients. If correctly computed, the value of x will fall in the range $-1 \leqslant x \leqslant +1$.

For the functions Apparent Sidereal Time at 0^h UT, Equation of the Equinoxes, Nutation in Longitude and Nutation in Obliquity, the variable t is measured in days of universal time (UT1 to be precise) from 0 January, 0^h UT. For all other functions in Section D, t is measured in days of terrestrial dynamical time (TDT) from 0 January, 0^h TDT. These latter functions can be evaluated for universal times, however, using the normalizing relation $x = ((t' + \Delta T) - W)/A - 1$, where t' is the universal time measured in days from 0 January, 0^h UT. As this volume goes to

Table 2: Comparison of *Almanac for Computers* and A^2
(High Precision Series, pp. D2-D29)

Function		No. of Terms	Span of Validity	Maximum Error
Apparent Sidereal Time at 0^h UT		34	95 days	0ˢ.001
Equation of the Equinoxes		"	"	0ˢ.001
Nutation in Longitude		"	"	0″.02
Nutation in Obliquity		"	"	0″.01
Sun:	Right Ascension	22	95 days	0ˢ.02
	Declination	"	"	0″.1
	Distance	"	"	4×10^{-7} AU
	Semidiameter	"	"	0″.01
	Ephemeris Transit	"	"	0ˢ.01
Moon:	Right Ascension	32	32 days	0ˢ.04
	Declination	"	"	0″.3
	Horizontal Parallax	"	"	0″.01
	Geocentric Rectangular Coords.	"	"	1×10^{-6} Earth radii
Mercury:	Right Ascension	32	95 days	0ˢ.08
	Declination	"	"	0″.3
	Distance	"	"	1×10^{-6} AU
Venus:	Right Ascension	32	95 days	0ˢ.02
	Declination	"	"	0″.6
	Distance	"	"	1×10^{-6} AU
Mars:	Right Ascension	16	95 days	0ˢ.03
	Declination	"	"	0″.2
	Distance	"	"	1×10^{-6} AU
Jupiter:	Right Ascension	16	95 days	0ˢ.03
	Declination	"	"	0″.2
	Distance	"	"	1×10^{-6} AU
Saturn:	Right Ascension	14	95 days	0ˢ.02
	Declination	"	"	0″.2
	Distance	"	"	3×10^{-6} AU
Uranus:	Right Ascension	14	95 days	0ˢ.1
	Declination	"	"	0″.2
	Distance	"	"	3×10^{-6} AU
Neptune:	Right Ascension	12	95 days	0ˢ.03
	Declination	"	"	0″.4
	Distance	"	"	1×10^{-5} AU
Pluto:	Right Ascension (astrometric)	12	95 days	0ˢ.004
	Declination (astrometric)	"	"	0″.04
	Distance	"	"	1×10^{-5} AU

press, $\Delta T = 54^s.2$ ($=0^d.000627$) appears to be a reliable value to use in 1984. Care should be taken to verify that the sum $t' + \Delta T$ falls within the time span for which the series is valid; if it falls outside, the series and constants for the next span should be used.

Once the normalized time variable x is determined, the series can be evaluated as follows:

let $\quad b_{n+1} = b_{n+2} = 0,$

compute $\quad b_i = 2xb_{i+1} - b_{i+2} + a_i,$ for $i = n, n-1, \ldots, 0,$

then $\quad f(x) = (b_0 - b_2)/2.$

Example: Compute the equation of the equinoxes to a precision of $\pm 0^s.05$ at $14^h\,51^m\,37^s$ UT ($=0^d.619178$) on 26 January 1984.

As shown in Table 3 the low precision series on page D30 provide the required precision. Since universal time is the independent variable for the series for the equation of the equinoxes,

$t = 26^d + 0^d.619178 = 26^d.619178$

Constants for the series are $A = 183.0$ and $W = 1$.

$x = (26.619178 - 1)/183.0 - 1 = -0.860004$

$b_{n+2} = b_{11} = 0$

$b_{n+1} = b_{10} = 0$

$b_n = b_9 = 2xb_{10} - b_{11} + a_9 = +0.0029$

$b_8 = 2xb_9 - b_{10} + a_8 = +0.0042$

$b_7 = 2xb_8 - b_9 + a_7 = -0.0321$

$b_6 = 2xb_7 - b_8 + a_6 = +0.0335$

$b_5 = 2xb_6 - b_7 + a_5 = +0.0275$

$b_4 = 2xb_5 - b_6 + a_4 = -0.0601$

$b_3 = 2xb_4 - b_5 + a_3 = +0.0787$

$b_2 = 2xb_3 - b_4 + a_2 = -0.0557$

$b_1 = 2xb_2 - b_3 + a_1 = +0.0572$

$b_0 = 2xb_1 - b_2 + a_0 = -1.9023$

$f(x) = (b_0 - b_2)/2 = (-1.9023 + 0.0557)/2$

equation of the equinoxes $= -0^s.92$

Beneath each set of coefficients is printed the sum of the coefficients. This may be used as an easy means of verifying the accuracy with which the coefficients have been entered in the computer.

The series for Apparent Sidereal Time are designed to reproduce the table 'Apparent Sidereal Time at 0^h Universal Time' in *The Astronomical Almanac*. To compute the Greenwich apparent sidereal time for any universal time,

Table 3: Comparison of *Almanac for Computers* and A^2
(Low Precision Series, pp. D30-D33)

Function		No. of Terms	Maximum Error
Apparent Sidereal Time at 0^h UT		10	$0^s.03$
Equation of the Equinoxes		"	$0^s.03$
Nutation in Longitude		"	$0''.5$
Nutation in Obliquity		"	$0''.3$
Sun:	Right Ascension	22	$0^s.5$
	Declination	"	$3''$
	Distance	"	4×10^{-5} AU
	Semidiameter	"	$0''.04$
	Ephemeris Transit	"	$0^s.5$
Mercury:	Right Ascension	50	8^s
	Declination	"	$2'$
	Distance	"	4×10^{-4} AU
Venus:	Right Ascension	50	$0^s.05$
	Declination	"	$0''.7$
	Distance	"	1×10^{-6} AU
Mars:	Right Ascension	34	$0^s.8$
	Declination	"	$5''$
	Distance	"	4×10^{-5} AU
Jupiter:	Right Ascension	34	$0^s.1$
	Declination	"	$0''.2$
	Distance	"	4×10^{-5} AU
Saturn:	Right Ascension	16	$0^s.1$
	Declination	"	$0''.6$
	Distance	"	4×10^{-5} AU
Uranus:	Right Ascension	16	$0^s.2$
	Declination	"	$0''.3$
	Distance	"	4×10^{-5} AU
Neptune:	Right Ascension	12	$0^s.1$
	Declination	"	$0''.5$
	Distance	"	7×10^{-5} AU
Pluto:	Right Ascension (astrometric)	12	$0^s.03$
	Declination (astrometric)	"	$0''.3$
	Distance	"	7×10^{-5} AU

(1) evaluate the series for the desired UT,

(2) add the desired UT to the result of step (1).

Local apparent sidereal time may be obtained by adding the local longitude to the Greenwich apparent sidereal time, where east longitudes are considered positive.

With two exceptions the series in Section D provide data referred to the true equinox and equator of date. The exceptions are

(1) the Moon's geocentric, rectangular coordinates (X, Y, Z), which are referred to the mean equator and equinox of B1950.0;

(2) the right ascension and declination of Pluto, which are astrometric (i.e., free of the effect of stellar aberration) and are referred to the mean equinox and equator of J2000.0.

The unit of distance for the Sun and planets is the astronomical unit; the unit of distance for the Moon is the Earth's equatorial radius.

Stellar Tables

The Stellar Tables (Section E) list the mean and apparent places of 176 stars for the current year, along with coefficients for converting from mean to apparent place for any date in the year. The selection of stars is essentially that of the star tables on pages 268–273 of the *Nautical Almanac*. Stars are arranged in order of increasing right ascension (decreasing sidereal hour angle), except where both components of a binary system are listed. For binary stars that can be resolved in small instruments, the position of one or both components is listed rather than the position of the center of gravity or the center of light. When both components of a binary system are included, the brighter star is listed first. For convenience of navigators the sidereal hour angle (SHA) is tabulated rather than right ascension (RA); right ascension in degrees can be obtained from the relation

$$RA = 360^\circ - SHA.$$

The quantities tabulated for each star are, from left to right on the page:

1. Identification number.
2. Navigational star number, provided the star is one of the 57 selected navigational stars listed in the *Nautical Almanac* and *Air Almanac*.
3. Star name. The Bayer designation is on the first line and the proper name, if any, is on the second line.
4. Magnitude and spectral type. The visual magnitude is on the first line, and the spectral type is on the second line. A composite spectrum is denoted by *.
5. Mean place of the star for J1984.5. The SHA in degrees is on the first line; the declination in degrees is on the second line.
6. Four coefficients (H, R, S, C) used in computing the apparent place of the star. The coefficients on the first line are for the computation of apparent SHA; these will hereafter be designated H_S, R_S, S_S, C_S. The coefficients on the second line are for the computation of apparent declination; these will be designated with the subscript D: H_D, etc.
7. The sum of the mean SHA or declination and the coefficients in the line. This may be used to verify that the numbers have been entered correctly in the computer.

The mean place of a star is a fundamental reference point with no simple geometric or observational significance. The apparent place of a star is the geocentric position, referred to the true equinox and equator of date, at which the star is observed. Thus the apparent place is the position needed for navigation, calibration of telescope setting circles, computation of transit times, etc. Except for Polaris the tabulated mean places for the middle of the year can be used to an accuracy of ±1′.3 for any date during the year. To obtain apparent places to greater accuracy, the following procedures should be used:

For the desired date in the current year determine τ, the fraction of the year elapsed. If t denotes the day of the year, τ can be computed from the relation $\tau = (t - W)/A$. As in previous editions the constants A and W are given at the top of page E3.

Except for Polaris, star positions accurate to better than ±0′.5 can be obtained from the following formulas:

$$\text{apparent SHA} = \text{mean SHA} + H_S + R_S\tau$$

$$\text{apparent decl.} = \text{mean decl.} + H_D + R_D\tau$$

Except for Polaris, star positions accurate to better than ±0′.1 (and usually better than ±0′.05) can be obtained from the following formulas:

$$\text{apparent SHA} = \text{mean SHA} + H_S + R_S\tau + S_S \sin(360^\circ\tau) + C_S \cos(360^\circ\tau)$$

$$\text{apparent decl.} = \text{mean decl.} + H_D + R_D\tau + S_D \sin(360^\circ\tau) + C_D \cos(360^\circ\tau)$$

To facilitate identification of the 57 standard navigational stars, an index for these stars is provided on page E2.

Example: Compute the apparent place of Regulus (α Leonis) on 5 August to an accuracy of ±0′.1.

From the calendar on pages A2-A3 or the formulas on pages B1-B2, 6 October is found to be day 218. From page E3, $A = 366.0$ and $W = 184$. Data for Regulus (Nav. No. 26; A/C ID 74) are found on page E6.

$\tau = (218 - 184)/366.0 = +0.0929$

	SHA	decl.
Mean place	208°.1131	+12°.0434
$+H$	+ 0.0039	+ 0.0020
$+R\tau$	− 0.0013	− 0.0004
$+S\sin(360^\circ\tau)$	+ 0.0023	+ 0.0008
$+C\cos(360^\circ\tau)$	+ 0.0029	+ 0.0011
Apparent place	208°.121	+12°.047

Because of the close proximity of Polaris to the north celestial pole, a small change in the position of Polaris on the celestial sphere causes a large change in the value of the SHA (or right ascension). This is purely due to the nature of the coordinate system rather than to extraordinary physical motion. Though the formulas given above will yield the declination of Polaris to an accuracy comparable to that of other stars, errors in SHA can reach ±1′.2, even if the more accurate formula is used.

Section B: APPLICATIONS

Introduction

In this section reference will be made to the following functions:

Sign function. The sign function serves to extract the algebraic sign from a number. The notation sign(x) is defined to be sign(x) = 1 for $x \geq 0$, sign(x) = -1 for $x < 0$. An equivalent definition is sign(x) = $x/|x|$ for $x \neq 0$, sign(x) = 1 for $x = 0$.
Examples: sign(247) = 1, sign(−6.28) = −1.

Truncation or largest-integer function. The truncation function extracts the integral part of a number. The algebraic sign of the result is the same as that of the original number. $\langle x \rangle$ is defined to be $\langle x \rangle = \text{sign}(x) \cdot N$, where N is the largest nonnegative integer such that $N \leq |x|$.
Examples: $\langle 17.835 \rangle = 17, \langle -3.1416 \rangle = -3$.

Modulus or remainder function. The modulus function yields the remainder of the division x/y, when the quotient is constrained to be an integral value. Thus mod(x,y) is defined to be $\text{mod}(x,y) = x - \langle x/y \rangle \cdot y$.
Examples: mod(11,3) = 2, mod(−764.3,360.0) = −44.3.
Note that $\langle x \rangle = x - \text{mod}(x,1.0)$. Therefore the truncation function can be defined in terms of the modulus function and *vice versa*. If either modulus or truncation is available on a calculator or computer, the other function can be simply obtained.

In this almanac universal time (UT) is to be identified with UT1, which is equivalent to the standard navigational time argument Greenwich Mean Time (GMT). The symbols UT and GMT may therefore be considered interchangable. For detailed information on time systems the reader should consult the Explanation of a current edition of *The Astronomical Almanac*.

Day of the Year

The day of the year (N) is defined as the integer $N = \langle t \rangle$, where t is the time elasped in days since 0 January of the current year. Thus N is an integer running from 1 through 365 (or 366 in leap years). The day of the year can be computed from either of the following formulas:

$$N = \left\langle \frac{275M}{9} \right\rangle - \left\langle \frac{M+9}{12} \right\rangle \left(1 + \left\langle \frac{K - 4\langle K/4 \rangle + 2}{3} \right\rangle\right) + I - 30$$

$$N = \left\langle \frac{275M}{9} \right\rangle - \left\langle \frac{M+9}{12} \right\rangle \left(1 + \left\langle \frac{\text{mod}(K,4) + 2}{3} \right\rangle\right) + I - 30$$

where N is the day of the year, K is the year (*e.g.*, 1981), M is the month ($1 \leq M \leq 12$), and I is the day of the month ($1 \leq I \leq 31$).

These formulas are equivalent and are valid for any year, except those centennial years that are not evenly divisible by 400. Therefore the formulas given above are

Sidereal Time

The following formulas are relevant to the computation of sidereal time:

(1) $GMST = 6^h.5905966 + 0^h.0657098242\,N + 1.00273791\,UT$

(2) $GMST = 6^h.69737456 + 2400^h.051336\,T_0 + 0^h.0000258622\,T_0^2 + 1.002737909\,UT$

(3) $\Omega = 74°.5658 - 0°.0529539\,(N + UT/24)$

(4) $\Omega = 125°.04452 - 1934°.13626\,T + 0°.002071\,T^2$

(5) $E = -0^h.00029 \sin\Omega$

(6) $GAST = GMST + E$

(7) $GAST = \Sigma(t_0) + 1.002737909\,UT = \Sigma(t) + UT$

(8) $LAST = GAST + \lambda/15$

where

GMST is the Greenwich mean sidereal time in hours;

Ω is the mean longitude of the ascending node of the Moon's orbit, measured in degrees;

E is the equation of the equinoxes in hours;

GAST is the Greenwich apparent sidereal time in hours;

LAST is the local apparent sidereal time in hours;

N is the day of the year ($1 \leqslant N \leqslant 365$ or, during a leap year, $1 \leqslant N \leqslant 366$);

T_0 and T are time intervals in Julian centuries from J2000.0:

$T_0 = (JD0 - 2451545.0)/36525 \qquad T = (JD - 2451545.0)/36525$;

UT is the universal time in hours;

JD0 and JD are the Julian Dates at 0^h UT and at an arbitrary time of the day, respectively;

$\Sigma(t_0)$ and $\Sigma(t)$ are values obtained by evaluating the Chebyshev series for Apparent Sidereal Time (pp. D2-D5 or D30) at 0^h UT and at an arbitrary time of the day, respectively; (see page A12 for notes about evaluating the series for sidereal time);

λ is the local longitude in degrees (east is positive; west is negative).

When using the formulas given above, it may be necessary to reduce the results to the range $0^h - 24^h$ by adding or subtracting multiples of 24^h.

Formulas (1) and (3) are specifically for the current year; the other formulas are valid at least over the latter half of this century. Formula (5) is an approximation that is accurate to about $\pm 0^s.2$. If more accuracy is required, the Chebyshev series for the Equation of the Equinoxes (pp. D2-D5 or D30) can be used in place of Formula (5). If sidereal time is to be computed to an accuracy better than $\pm 0^s.2$ (rarely justified for practical applications), then *either* the Chebyshev series for the Equation of the Equinoxes should be used in place of Formula (5) *or* Formula (7) should be used in place of Formula (6).

Hour Angles

The following formulas are useful if astronomical data, such as that given in Sections C and E, are applied to navigational purposes:

$$GHA = 15\,(GAST - RA)$$

$$LHA = 15\,(LAST - RA) = GHA + \lambda$$

$$GHA\ Aries = 15\ GAST$$

$$SHA = 360^\circ - 15\ RA$$

$$GHA = GHA\ Aries + SHA$$

where

GHA is the Greenwich hour angle in degrees;
LHA is the local hour angle in degrees;
GHA Aries is the Greenwich hour angle of the First Point of Aries (the origin of right ascension) in degrees;
SHA is the sidereal hour angle in degrees;
RA is the apparent right ascension (referred to the true equator and equinox of date) in hours;
λ is the local longitude in degrees (east is positive; west is negative);
GAST is the Greenwich apparent sidereal time in hours;
LAST is the local apparent sidereal time in hours.

When using the above formulas, it may be necessary to add or subtract 360° to reduce the resulting hour angles to the range 0° –360°. Often the local hour angle values are reduced to the range –180° to +180°, in which case they are called meridian angles. In all cases positive hour angle values are measured westward from the meridian.

Altitude and Azimuth

The following formulas can be used to compute the altitude (a) and azimuth (A) of a celestial body:

(1) $\sin a = \cos z = \sin\phi \sin\delta + \cos\phi \cos\delta \cos LHA$

(2) $x = \tan A = \sin LHA / (\cos LHA \sin\phi - \tan\delta \cos\phi)$

Since computers and calculators normally give the arctangent in the range –90° to + 90°, the correct quadrant for A can be selected according to the following rules:

If $0^\circ \leqslant LHA \leqslant 180^\circ$,

$A = 180^\circ + \arctan x$, if x is positive,
$A = 360^\circ + \arctan x$, if x is negative.

If $180^\circ \leqslant LHA < 360^\circ$,

$A = \arctan x$, if x is positive,
$A = 180^\circ + \arctan x$, if x is negative.

Notation:

a = altitude of body above (if $\sin a > 0$) or below (if $\sin a < 0$) the horizon;
A = azimuth of body measured eastward from north over the range $0° \leqslant A \leqslant 360°$;
ϕ = latitude of observer (north is positive; south is negative);
δ = declination of body (north is positive; south is negative);
LHA = local hour angle of body;
z = zenith distance of body ($z = 90° - a$).

In standard navigational notation altitude and azimuth are denoted Hc and Zn, respectively. Equations (1) and (2) are the basic formulas used in preparing sight reduction tables; they do not include the effect of refraction.

Example: Compute the altitude and azimuth of the Sun at $17^h58^m45^s$ UT on 24 May 1984 at Annapolis, Maryland.

Latitude: $\phi = +38.^\circ59$ $\sin\phi = +0.62374$ $\cos\phi = +0.78163$
Longitude: $\lambda = -76.^\circ30$
Using the power series on page C4, the Sun's GHA and δ are found to be
GHA = 90.°483 hence LHA = 90.°483 − 76.°30 = 14.°183
$\sin$LHA = +0.24502 $\cos$LHA = +0.96952
$\delta = +20.^\circ899$ $\sin\delta = +0.35672$ $\cos\delta = +0.93421$ $\tan\delta = +0.38184$
$\sin a = \cos z = (0.62374)(0.35672) + (0.78163)(0.93421)(0.96952)$
$= +0\ 93045$
$a = 68.^\circ5$
$x = \tan A = 0.24502/((0.96952)(0.62374) - (0.38184)(0.78163))$
$= +0.80001$ $\arctan x = +38.^\circ7$
Since LHA is greater than 0° and less than 180°, and since x is positive,
$A = 180° + 38.^\circ7 = 218.^\circ7$

Sunrise, Sunset and Twilight

For locations between latitudes 65° North and 65° South, the following algorithm provides times of sunrise, sunset and twilight to an accuracy of $\pm 2^m$, for any date in the latter half of the twentieth century. Because the phenomena depend on local meteorological conditions, attempts to attain higher accuracy are seldom justified. Although the algorithm can be used at higher latitudes, its accuracy deteriorates near dates on which the Sun remains above or below the horizon for more than twenty-four hours.

Notation:

ϕ = latitude of observer (north is positive; south is negative)
λ = longitude of observer (east is positive; west is negative)
M = Sun's mean anomaly
L = Sun's true longitude

RA = Sun's right ascension
δ = Sun's declination
H = Sun's local hour angle
z = Sun's zenith distance at rise, set or twilight*
t = approximate time of phenomenon in days since 0 Jan., 0^h UT
T = local mean time of phenomenon
UT = universal time of phenomenon

*The proper value of z should be chosen from the following:

	z	$\cos z$
Sunrise and Sunset	90°50′	−0.01454
Civil Twilight	96°	−0.10453
Nautical Twilight	102°	−0.20791
Astronomical Twilight	108°	−0.30902

Formulas:

(1) $M = 0.^\circ 985600t - 3.^\circ 289$
(2) $L = M + 1.^\circ 916 \sin M + 0.^\circ 020 \sin 2M + 282.^\circ 634$
(3) $\tan RA = 0.91746 \tan L$
(4) $\sin\delta = 0.39782 \sin L$
(5) $x = \cos H = (\cos z - \sin\delta \sin\phi) / (\cos\delta \cos\phi)$
(6) $T = H + RA - 0.^h 065710t - 6.^h 622$
(7) $UT = T + \lambda$

Procedure:

1. With an initial value of t, compute M from Eq. (1) and then L from Eq. (2). If a morning phenomenon (sunrise or the beginning of morning twilight) is being computed, construct an initial value of t from the formula
$$t = N + (6^h - \lambda)/24$$
where N is the day of the year (see the calendar on pages A2–A3 or the formulas on page B1) and λ is the observer's longitude expressed in hours. If an evening phenomenon is being computed, use
$$t = N + (18^h - \lambda)/24$$
2. Solve Eq. (3) for RA, noting that RA is in the same quadrant as L. Transform RA to hours for later use in Eq. (6).
3. Solve Eq. (4) for $\sin\delta$ which appears in Eq. (5); $\cos\delta$, which also is required in Eq. (5), should be determined from $\sin\delta$. While $\sin\delta$ may be positive or negative, $\cos\delta$ is always positive.
4. Solve Eq. (5) for H. Since computers and calculators normally give the arccosine in the range 0° −180°, the correct quadrant for H can be selected according to the following rules:
   (a) rising phenomena, $H = 360^\circ - \arccos x$;
   (b) setting phenomena, $H = \arccos x$.

In other words, for rising phenomena H must be either in quadrant 3 or 4 (depending on the sign of $\cos H$), whereas H must be either in quadrant 1 or 2 for setting phenomena. Convert H from degrees to hours for use in Eq. (6).

5. Compute T from Eq. (6), recalling that H and RA must be expressed hours. If T is negative or greater than 24^h, it should be converted to the range $0^h - 24^h$ by adding or subtracting multiples of 24^h.
6. Compute UT from Eq. (7), where λ must be expressed in hours. UT is an approximation to the time of sunrise, sunset or twilight, referred to the Greenwich meridian. If UT is greater than 24^h, the phenomenon occurs on the following day, Greenwich time. If UT is negative, the phenomenon occurs on the previous day, Greenwich time.

To ensure that precision is not lost during the computations, t should be carried to four decimal places. Angles should be expressed to three decimals of a degree and, upon conversion, to three decimals of an hour. Five significant digits should be carried for the trigonometric functions.

Under certain conditions Eq. (5) will yield a value of $|\cos H| > 1$, indicating the absence of the phenomenon on that day. At far northern latitudes, for example, there is continuous illumination during certain summer days and continuous darkness during winter days.

Example: Compute the time of sunrise on 25 June at Wayne, New Jersey.

Latitude: 40°.9 North Longitude: 74°.3 West

$\phi = +40°.9$ $\sin\phi = +0.65474$ $\cos\phi = +0.75585$

$\lambda = -74°.3/15 = -4^h.95$

For sunrise: $z = 90°50'$ $\cos z = -0.01454$

$t = 176^d + (6^h + 4^h.95)/24 = 176^d.456$

$M = 0°.985600\,(176^d.456) - 3°.289 = 170°.626$

$L = 170°.626 + 1°.916\,(0.16288) + 0°.020\,(-0.32141) + 282°.634$
$= 453°.566 = 93°.566$

$\tan RA = 0.91746\,(-16.047) = -14.723$

$RA = 93°.886/15 = 6^h.259$ Since L is in quadrant 2, so is RA.

$\sin\delta = 0.39782\,(0.99806) = 0.39705$

$\cos\delta = 0.91780$

$x = \cos H = [-0.01454 - (0.39705)(0.65474)]/[(0.91780)(0.75585)]$
$= -0.39570$ $\arccos x = 113°.310$

Since sunrise is being computed, $H = 360° - 113°.310 = 246°.690$

$H = 246°.690/15 = 16^h.446$

$T = 16^h.446 + 6^h.259 - 0^h.065710\,(176^d.456) - 6^h.622 = 4^h.488$

$UT = 4^h.488 + 4^h.95 = 9^h.44$

Sunrise occurs at 9^h26^m UT = 5^h26^m EDT

B8

Solar Coordinates

The true geocentric longitude of the Sun (L) can be computed to an accuracy of ±1 minute of arc from the following formulas:

$$M = 357.^\circ528 + 35999.^\circ050T$$
$$\mathcal{L} = 280.^\circ460 + 36000.^\circ772T$$
$$L = \mathcal{L} + (1.^\circ915 - 0.^\circ0048T)\sin M + 0.^\circ020 \sin 2M$$

where $T = (\text{JD} - 2451545.0)/36525$ and JD is the Julian Date (see page B2).

If we consider the Sun's latitude to be identically zero, the right ascension (RA) and declination (δ) of the Sun can also be computed to ±1 minute of arc from

$$\tan RA = \cos\epsilon \tan L$$
$$\sin\delta = \sin\epsilon \sin L$$

where ϵ, the obliquity of the ecliptic, can be computed from $\epsilon = 23.^\circ439 - 0.^\circ013T$. The right ascension is always in the same quadrant as the true longitude.

Because the obliquity varies slowly, a single value can be used for an extended period of time. During the last quarter of the twentieth century, $\epsilon = 23.^\circ441$ is sufficiently accurate. Similarly the coefficient of $\sin M$ in the equation for L changes slowly; for the last half of the twentith century a value of $1.^\circ916$ can be safely used.

Although there is no rigorous limit on the time span for which these formulas are valid, their accuracy gradually deteriorates for values of T greater than a couple of centuries.

Equation of Time and Time of Solar Transit

The equation of time (EqT) is the hour angle of the true Sun minus the hour angle of the mean sun. Thus it is the difference: apparent solar (sundial) time minus mean solar (clock) time.

For the current year EqT can be computed to an accuracy of ±0.8 minute from the following formula:

(1) $$EqT = -7.^m64 \sin(0.^\circ9856\,t) + 0.^m57 \cos(0.^\circ9856\,t) - 9.^m36 \sin(1.^\circ9712\,t) - 2.^m84 \cos(1.^\circ9712\,t)$$

where t is the number of days since 0 January, 0^h UT.

If higher accuracy is required the following formulas will give EqT to an accuracy of ±2 seconds during the current year:

(2) $$\theta = 8.^\circ855 + 0.^\circ98561\,t + 1.^\circ916 \sin(0.^\circ9856\,t - 3.^\circ819) + 0.^\circ020 \sin(1.^\circ9712\,t - 7.^\circ638)$$

(3) $$EqT = 35.^m421 + 3.^m94244\,t - 4.^m0 \arctan[(\tan\theta)/0.91747]$$

where t is the number of days, and fractions thereof, since 0 January, 0^h UT. In Eq. (3) the arctangent should yeild a result in degrees that is in the same quadrant as θ. Near the end of the year θ becomes greater than 360°. When this occurs the arctangent in Eq. (3) should also be greater than 360°.

Eqs. (2) and (3) can be used to compute the time at which the Sun transits the local meridian. First use Eqs. (2) and (3) to compute EqT for $t = N + (12^h - \lambda)/24$, where N is the day of the year (see the calendar on pages A2-A3 or the formulas on pages B1-B2) and λ is the longitude (east positive, west negative) expressed in hours. Then the local mean time (LMT) of transit is given to an accuracy of ±2 seconds by $LMT = 12^h - EqT$. The universal time of local transit is then obtained from $UT = LMT - \lambda$.

Example: Compute the time of solar transit at longitude 9°07'2 East on 13 December 1984.

$\lambda = +9.^\circ127/15 = +0.^h6080 = +0^h36.^m48$

For solar transit: $t = 348^d + (12^h - 0.^h60805)/24 = 348.^d4747$

$\theta = 8.^\circ855 + 0.^\circ98561\,(348.^d4747) + 1.^\circ916\,(-0.3480)$
$+ 0.^\circ020\,(-0.6524) = 351.^\circ635$

$EqT = 35.^m421 + 3.^m94244\,(348.^d4747) - 4.^m0 \arctan[-0.14704/0.91747]$
$= 35.^m421 + 1373.^m841 - 4.^m0(350.^\circ895) = +5.^m68$

$LMT = 12^h00^m - 5.^m68 = 11^h54.^m32$

$UT = 11^h54.^m32 - 0^h36.^m48 = 11^h17^m50^s$ UT

Moonrise and Moonset

Times of moonrise and moonset can be computed for specified locations using the following algorithm. Between latitudes 60° North and 60° South, the phenomena can be computed to an accuracy of ±5^m. Although the algorithm can be used at higher latitudes, its accuracy deteriorates near dates on which the Moon remains above or below the horizon for more than twenty-four hours.

Notation:
- ϕ = latitude of observer (north is positive; south is negative)
- λ = longitude of observer (east is positive; west is negative)
- t_i = *i-th* approximation to universal time of phenomenon, expressed in fractions of a day from 0^h UT
- GHA_i = Moon's GHA at time t_i
- δ_i = Moon's declination at time t_i (north is positive; south is negative)
- τ_i = *i-th* correction to t_0, thus $t_i = t_0 + \tau_i$
- H_i = *i-th* approximation to Moon's LHA at time of rise or set
- ΔH_i = *i-th* approximation to Moon's daily rate of change in GHA

Formulas:

(1) $\Delta H_i = (GHA_i - GHA_0) / \tau_i$ for $i = 0$, let $\Delta H_0 = 347.^\circ81$

(2) $x_{i+1} = \cos H_{i+1} = (.00233 - \sin\phi \sin\delta_i) / (\cos\phi \cos\delta_i)$

(3) $\tau_{i+1} = (H_{i+1} - H_0) / \Delta H_i$

(4) $t_{i+1} = t_0 + \tau_{i+1}$

Procedure:

1. Let $t_0 = (12^h - \lambda)/24$, where λ is the observer's longitude expressed in hours. Set $i = 0$ and begin the following iterative process.
2. For time t_i compute the Moon's GHA and declination to navigational precision ($\pm 0'.1$). Label these quantities GHA_i and δ_i, respectively, where i specifies the iteration number. For $i = 0$, compute $H_0 = GHA_0 + \lambda$.
3. If $i = 0$, let $\Delta H_0 = 347°.81$. Otherwise compute ΔH_i from Eq. (1). If $GHA_i < GHA_0$, add 360° to GHA_i before computing ΔH_i.
4. Solve Eq. (2) for H_{i+1}. Since computers and calculators normally give the arccosine in the range 0° - 180°, the correct quadrant for H_{i+1} can be selected according to the following rules:
   (a) moonrise computations, $H_{i+1} = 360° - \arccos x_{i+1}$;
   (b) moonset computations, $H_{i+1} = \arccos x_{i+1}$.
   In other words, near the time of moonrise H_{i+1} must be either in quadrant 3 or 4 (depending on the sign of $\cos H_{i+1}$); near moonset H_{i+1} must be either in quadrant 1 or 2. For latitudes higher than 60° (i.e., $|\phi| > 60°$), the condition $|\cos H_{i+1}| > 1$ can occur, thereby indicating the absence of the phenomenon on that day.
5. Compute τ_{i+1} from Eq. (3). If $|\tau_{i+1}| < 0^d.5$, proceed to Step 6. If $|\tau_{i+1}| > 0^d.5$, the phenomenon being computed occurs on the day prior to the day desired (if τ_{i+1} is negative) or on the day following the day desired (if τ_{i+1} is positive). Normally the phenomenon on the desired day can be obtained by adding to τ_{i+1} (if τ_{i+1} is negative), or subtracting from τ_{i+1} (if τ_{i+1} is positive), 360°, ΔH_i. If successful this technique will produce a new value of τ_{i+1} in the required range. However, two conditions may prevent the reduction to $|\tau_{i+1}| < 0^d.5$:
   (a) for low values of i, τ_{i+1} may be a fairly crude approximation to the ultimate value, τ_n;
   (b) each month there is one day (near last quarter) on which there is no moonrise, and another day (near first quarter) on which there is no moonset.
   If $|\tau_{i+1}| \approx 0^d.5$, it is probably worth attempting another iteration to see if $|\tau_{i+2}| < 0^d.5$.
6. Compute t_{i+1} from Eq. (4). If $|t_{i+1} - t_i| < 0^d.01$, t_{i+1} is accurate to $\pm 5^m$. Otherwise it is necessary to iterate the solution by setting $i = i + 1$ and executing Steps 2 through 6 again.

Example: Compute moonrise on 15 October 1984 at San Francisco, California.

$\phi = +37°.45$ $\sin\phi = +0.60807$ $\cos\phi = +0.79388$

$\lambda = -122°.27 = -8^h.151$

$t_0 = (12^h + 8^h.151)/24 = 0^d.83963$

$i = 0$: Evaluating the power series on page C17 for t_0 on 15 October,

$GHA_0 = 239°.075$ $\delta_0 = +26°.045$

$H_0 = 239°.075 - 122°.270 = 116°.805$

$\Delta H_0 = 347°.81$

$x_1 = \cos H_1 = [0.00233 - (0.60807)(0.43908)] / [(0.79388)(0.89845)]$

$= -0.37106$ $\arccos x_1 = 111°.781$

Since moonrise is sought, H_1 is in quadrant 3 or 4:

$H_1 = 360° - 111°.781 = 248°.219$

$\tau_1 = (248°.219 - 116°.805)/347°.81 = +0^d.37783$

$|\tau_1| < 0^d.5$ as required.

$t_1 = 0^d.83963 + 0^d.37783 = 1^d.21746 = 5^h\,13^m$ UT on 16 October

$i = 1$: Evaluating the power series on page C17 for t_1 on 16 October,

$GHA_1 = 10°.087$ $\delta_1 = +26°.402$

$\Delta H_1 = (370°.087 - 239°.075)/0^d.37783 = 346°.749$

$x_2 = \cos H_2 = [0.00233 - (0.60807)(0.44467)] / [(0.79388)(0.89570)]$

$= -0.37698$ $\arccos x_2 = 112°.147$

Since moonrise is sought, H_2 is in quadrant 3 or 4:

$H_2 = 360° - 112°.147 = 247°.853$

$\tau_2 = (247°.853 - 116°.805)/346°.749 = +0^d.37793$

$t_2 = 0^d.83963 + 0^d.37793 = 1^d.21756 = 5^h\,13^m$ UT on 16 October

$=$ 10:13 p.m., Pacific Daylight Time, on 15 October

$|t_2 - t_1| = 0^d.0001 < 0^d.01$

The extremely rapid convergence illustrated in this example occurs frequently but not invariably. Although the first approximation (t_1) will often give adequate precision for most purposes, it is recommended that the solution be iterated and that the convergence criterion ($|t_{i+1} - t_i| < 0^d.01$) be tested.

Polaris (Pole Star)

The following formulas are relevant to observations of Polaris:

(1) $\phi = a - p\cos h + 0.5p \sin p \sin^2 h \tan\phi$

(2) $A\cos\phi = -p\sin h - p\sin p \sin h \cos h \tan\phi$

where p is the polar distance of Polaris: $p = 90° -$ declination of Polaris

h is the LHA of Polaris: $h =$ GHA Aries + SHA Polaris + east ($-$west) longitude of observer

ϕ is the observer's latitude;

A is the azimuth of Polaris;

a is the corrected altitude of Polaris.

Eq. (1) permits the observer's latitude to be determined from an observation of the altitude of Polaris (corrected for refraction, dip, etc.). Assumed values of the observer's latitude and longitude can be used for the right side of Eq. (1). Eq. (2) yields the azimuth of Polaris if the observer's position is known. These expressions are accurate only for Polaris, since they depend on p being a small quantity. The SHA and declination of Polaris to be used in these formulas should be referred to the true equator and equinox of date; i.e., the apparent place of Polaris should be computed (see Section E 'Stellar Tables'; Polaris is star number 17).

Equation of Position Line

The following formula can be used to obtain a line of position (LOP) directly from an observation of the altitude of a celestial body:

$$\lambda = \text{GHA} \pm \arccos\left[(\sin a - \sin\phi \sin d) / \cos\phi \cos d\right]$$

where: λ is the computed longitude;
GHA is the GHA of the body for the time of observation;
a is the corrected altitude of the body;
d is the declination of the body for the time of observation;
ϕ is an estimate of the observer's latitude.

North latitudes and west longitudes are positive; south latitudes and east longitudes are negative. Longitudes with absolute values greater than 180° may be encountered. In the above formula, + is used for bodies east of the meridian (rising) and − for bodies west of the meridian (setting).

The formula gives the longitude λ at which the position line crosses the parallel of latitude ϕ. Repeated application of the formula using different values of latitude yields a locus of points all lying in the LOP. Note that no assumed position is necessary, although an estimate of the observer's latitude is helpful in reducing the number of times the formula is applied.

The formula becomes indeterminate at the transit time of a body and for latitudes that the position line does not cross at any point.

Motion of Body and Motion of Observer

During the time interval Δt (e.g., the interval between a sextant observation and the time of a fix), the rotation of the Earth causes a change in the altitude of a celestial body. To permit the use of a common assumed position and LHA Aries for observations made at different times, the following correction can be applied to the observed altitude:

$$\text{MOB} = 15.04\, \Delta t \cos\phi \sin A$$

where MOB is the altitude correction in minutes of arc, Δt is the time difference in minutes, ϕ is the latitude of the observer, and A is the azimuth of the observed body. If the time of the fix is later than the time of observation, MOB should be added to the observed altitude. It should be noted that the formula for MOB is an approximation that becomes unreliable for values of Δt greater than 5 minutes.

The following formula gives the change of altitude of a celestial body due to the motion of the observer in the time interval Δt (e.g., the interval between a sextant observation and the time of a fix). Though this formula is only an approximation to the physical phenomenon, it is the exact mathematical equivalent of advancing or retiring a line of position.

$$\text{MOO} = \frac{v\,\Delta t}{60} \cos (A - C)$$

where MOO is the altitude correction in minutes of arc, Δt is the time difference in minutes, A is the azimuth of the observed body, C is the track/course angle, and v is the ground speed in knots. If the time of the fix is later than the time of observation, MOO should be added to the observed altitude.

Sextant Altitude Corrections

Several corrections must be applied to a sextant altitude (hs) in order to obtain a corrected altitude (Ho). Ho can then be either (a) compared with the computed altitude (Hc) to obtain the altitude difference (Δa); or (b) used in the 'Equation of Position Line' (see p. B12) to obtain directly the location of the LOP for the sight.

The corrections, in the order in which they should be applied, are:

(1) Instrument and/or index correction, IC;
(2) Dip of Horizon, D (marine sextant); or Coriolis correction, Δz (bubble sextant);
(3) Atmospheric refraction, R;
(4) Semidiameter, SD (marine sextant, Sun and Moon observations);
(5) Parallax in altitude, PA (Moon, Venus and Mars observation).

In mathematical notation:

$$\text{Ho} = \text{hs} + \text{IC} + (\text{D or } \Delta z) - \text{R} + \text{SD} + \text{PA}$$

If Venus is observed, an additional correction for the phase of the planet may be necessary. This correction can be made either to the sextant altitude or to the GHA or LHA of Venus.

Descriptions and formulas for D, Δz, R, SD, PA and the phase correction for Venus are given on the following pages.

Dip of Horizon

The dip of the apparent horizon from a horizontal plane is given by

$$\text{D} = -0.97\sqrt{h}$$

where h is the height of eye level of the observer in feet and D is the dip of the horizon in minutes of arc. For observations of a celestial body made with a marine sextant or similar instrument, D should be added to the observed altitude to obtain the corrected altitude. This formula is an approximation; the apparent dip varies with atmospheric conditions.

Coriolis Correction

Any object moving across or above the surface of the rotating Earth is subject to an apparent force tending to push the object to the right in the northern hemisphere and to the left in the southern hemisphere. This Coriolis acceleration manifests itself as a deflection of the apparent vertical by an amount Z:

(1) $$Z = 2.62\,V\sin\phi + 0.146\,V^2\sin C\tan\phi - 5.25\,VC'$$

where: Z is the deflection in minutes of arc;
V is the speed in hundreds of knots;
ϕ is the latitude;
C is the true track/course angle;
C' is the rate of change of true track/course angle in degrees per minute of time.

The 'Coriolis (Z) Correction' tabulated in the Air Almanac consists of only the first term in Eq. (1). The second term is known as 'Rhumb Line Correction', and the third term is the 'Wander Correction'. Usually only the first term is significant.

Observations of the altitudes of celestial bodies made with bubble sextants or similar artificial horizon instruments must be corrected for the Coriolis effect. The correction Δz, which can be added to the observed (*e.g.*, bubble sextant) altitude, is given approximately by

(2) $$\Delta z = Z\sin(A - C)$$

where: Δz is the altitude correction in minutes of arc;
Z is the deflection of the vertical determined from Eq. (1);
A is the azimuth of observed body;
C is the true track/course angle.

In the northern hemisphere the correction Δz is positive for stars on the right and negative for stars on the left of the aircraft. In the southern hemisphere the correction is negative for stars on the right and positive for stars on the left.

Atmospheric Refraction

The Earth's atmosphere tends to refract light in such a way that celestial bodies appear slightly higher in the sky than they would if there were no atmosphere. The formulas below can be used to determine R, the angle of refraction. R should be subtracted from an observed (e.g., sextant) altitude to obtain the corrected altitude.

(1) $$R = \frac{P}{273+T}\,[3.430289\,(z - \arcsin[0.9986047\sin(0.9967614z)]) - 0.01115929z]$$

(2) $$R = \exp(-h/27000)\tan z = 1/[\exp(h/27000)\tan a]$$

where: R is the refraction correction in minutes of arc;
a is the observed altitude;
z is the observed zenith distance in degrees: $z = 90° - a$;

T is the temperature in degrees Celsius;
P is the atmospheric pressure in millibars;
h is the height of the observer above sea level in feet.

The formulas given above are approximations and are not equivalent to a complete theory of refraction. Eq. (1), which is better suited to surface observations, is accurate to ±0′.1 for altitudes greater than 15°; between altitudes 3° and 15°, errors can reach ±1′.0; for altitudes less than 3°, errors between ±1′ and ±3′ will be encountered. Eq. (2), which is better suited to observations from aircraft, should only be used for altitudes greater than 10°. Above 10° this formula is accurate to ±0′.2.

For surface observations under standard atmospheric conditions, the following Chebyshev series represents refraction for altitudes from 0° to 90° with errors not exceeding ±0′.1.

$$R = a_0/2 + \sum_{i=1}^{9} a_i T_i(x)$$

where x is related to the observed altitude by $x = 0.442837 \log_e(a + 1.5) - 1$. The coefficients a_i in the series are

$a_0 = +28.891741$	$a_5 = +0.340097$
$a_1 = -20.516167$	$a_6 = -0.024576$
$a_2 = +\ 7.291562$	$a_7 = -0.050041$
$a_3 = -\ 0.813492$	$a_8 = +0.023252$
$a_4 = -\ 0.690042$	$a_9 = -0.009406$

The sum of these coefficients is +14.442928.

For a given value of the altitude a, compute x. Then the series can be evaluated as follows:

let $b_{10} = b_{11} = 0$,
compute $b_i = 2x b_{i+1} - b_{i+2} + a_i$, for $i = 9, 8, \ldots, 0$,
then $R = (b_0 - b_2)/2$.

Example: A star is observed from the Earth's surface under standard atmospheric conditions to be at altitude 10°.0. Use the Chebyshev series to compute the refraction correction.

$a = 10°.0$ $\quad x = 0.442837 \log_e(10.0 + 1.5) - 1 = +0.081562$

$b_{10} = b_{11} = 0$

$b_9 = 0.0 - 0.0 - 0.009406 = -0.009406$
$b_8 = -0.001534 - 0.0 + 0.023252 = +0.021718$
$b_7 = +0.003543 + 0.009406 - 0.050041 = -0.037092$
$b_6 = -0.006051 - 0.021718 - 0.024576 = -0.052345$
$b_5 = -0.008539 + 0.037092 + 0.340097 = +0.368650$
$b_4 = +0.060136 + 0.052345 - 0.690042 = -0.577561$
$b_3 = -0.094214 - 0.368650 - 0.813492 = -1.276356$
$b_2 = -0.208204 + 0.577561 + 7.291562 = +7.660919$
$b_1 = +1.249680 + 1.276356 - 20.516167 = -17.990131$
$b_0 = -2.934622 - 7.660919 + 28.891741 = +18.296200$

$$R = (18.296200 - 7.660919)/2 = 5′.3$$

Semidiameter of the Sun and Planets

The semidiameters of the Sun and planets can be computed from

$$SD = S/d = S\pi/8.794$$

where: SD is the semidiameter in seconds of arc;
S is the semidiameter at unit distance (1 AU) in seconds of arc;
d is the geocentric distance in AU;
π is the horizontal parallax in seconds of arc.

The following values of S should be used:

Sun	959″.63	Jupiter	98″.47
Mercury	3.34	Saturn	83.33
Venus	8.41	Uranus	34.28
Mars	4.68	Neptune	36.56

These values apply to the equatorial dimensions of the bodies and do not include any adjustments for irradiation.

Semidiameter of the Moon

The geocentric semidiameter of the Moon can be computed from

(1) $$SD = 56204.92/d = 0.272476\,\pi$$

where SD is the geocentric semidiameter in seconds of arc, d is the geocentric distance of the Moon in units of the Earth's equatorial radius, and π is the horizontal parallax of the Moon in seconds of arc.

Since observations are made from the Earth's surface rather than from its center, the observed, topocentric semidiameter is slightly greater than the geocentric semidiameter. For navigation and certain other purposes the *augmented semidiameter* of the Moon should be used:

(2) $$SD_{aug} = SD[1 + (\sin a)/d]$$

where SD_{aug} is the augmented semidiameter in seconds of arc, a is the altitude of the Moon (for navigational purposes a = Ho, but hs or Hc can be used instead with negligible error), d is the geocentric distance of the Moon in units of the Earth's equatorial radius, and SD is the geocentric semidiameter computed from Eq.(1). For navigational purposes a constant value of $d = 60.27$ can be used to sufficient accuracy. The increase in the Moon's semidiameter due to augmentation is zero when the Moon is on the horizon and is about 0′.3 when the Moon is at the zenith.

Parallax in Altitude

The finite size of the Earth causes a parallactic shift in the apparent positions of nearby celestial objects. The resulting parallax in altitude can be computed from

$$\sin PA = \sin\pi \cos a$$

where PA is the parallax in altitude, π is the horizontal parallax, and a is the observed altitude. When the horizontal parallax of a body is not available, it can be computed from the relation $\pi = 8''.794/d$, where d is the geocentric distance of the body in astronomical units. Except for the Moon, parallax in altitude does not exceed $1'$. Since parallax tends to decrease the apparent altitude of a body, the quantity PA should be added to an observed (e.g., sextant) altitude in order to obtain the corrected altitude. To a reasonable approximation, PA can also be computed from

$$\mathrm{PA} = \pi \cos a$$

Correction for the Phase of Venus

When the altitude of Venus is observed with a small instrument, a correction to the observed altitude is required to account for the fact that the center of light, rather than the center of the disk, is observed. This correction has the form $-k \cos\theta$, where k is a correction factor (given below) and θ is the angle on the celestial sphere, at the position of Venus, between the observer's vertical and the direction of the Sun. The correction, which should be added to the observed (e.g., sextant) altitude, is positive when the Sun is lower than Venus, zero when they have the same altitude, and negative when the Sun is higher.

In sight reduction this effect can be approximately taken into account by correcting the GHA (or LHA) of Venus rather than correcting the observed altitude. Simply add k to the GHA (or LHA) of Venus when Venus is east of the Sun (i.e., when Venus is an evening planet), and subtract k from the GHA (or LHA) when Venus is west of the Sun (morning planet). The correction should not be applied in this way near the time of superior or inferior conjunction.

In 1984 Venus is in the morning sky from the beginning of the year until mid-May, when it becomes too close to the Sun for observation. Late in May it reappears in the evening sky, where it is visible for the rest of the year.

The values of k for 1984 are

		k
Jan.	1	0.1
Dec.	12	0.2
Dec.	31	

```
10 ! MOON-1984.TEMP    MOONRISE/MOONSET/MOON PHASE (NAVAL OBSERVATORY, VALUES GOOD THRU CY 1984)
20 ! VALID FOR LOCATIONS 60degN THRU 60degS LATITUDE
30 COM B(20),C,X,W$[60],T,W,G,E,V,S,P$[1],R ,INTEGER D(192)
40 GCLEAR @ MSCALE 0,0 @ CSIZE 9 @ MOVE 4,55 @ LABEL "MOONRISE/MOONSET/MOON PHASE"
50 MOVE 45,49 @ LABEL "PROGRAM"
60 CSIZE 5 @ MOVE 17,43 @ LABEL "(VALID FOR LATITUDES 60degN TO 60degS)"
70 MOVE 33,9 @ LABEL "PRESS 'CONT' WHEN READY..." @ PAUSE
80 GCLEAR
90 PRINTER IS 506,132 @ DEG @ DIM D$[100]
100 PRINT " &11V";" &k0S" @ FOR I=1 TO 25 @ PRINT "MOON-";@ NEXT I @ PRINT "MOON";" &17V"
110 CLEAR @ PRINT " &k3S",TAB (32);"MOONRISE/MOONSET/MOON PHASE--198[illegible]"
120 PRINT TAB (27);"_____"
130 PRINT " &k0S" @ PRINT TAB (38);"Nautical Almanac Office, US Naval Observatory" @ PRINT @ PRINT
140 PRINT TAB (42);"(VALID FOR LATITUDES 60degN TO 60degS)" @ PRINT " &15V"
150 C=1 @ Q3=0 @ Q2=0
160 CLEAR @ D$=" ENTER THE MONTH, DAY (example: 10,28)" @ DISP HGL?$ (D$,1) @ INPUT MONTH,DAY
170 YEAR=1984 ! VALUE TO BE UPDATED ANNUALLY
180 IF MONTH>12 OR DAY>31 THEN BEEP @ CLEAR @ GOTO 160
190 PRINT " &k3S"
200 PRINT USING 210 ; MONTH,DAY,YEAR
210 IMAGE "DATE: ",DD,"/",ZZ,"/",DDDD
220    IF YEAR/4=INT (YEAR/4) THEN RESTORE 240 ELSE RESTORE 250 ! DETERMINES LEAP YEAR
230    ! LINE 240 = LEAP YEAR (366 DAYS)    LINE 250 = 365-DAY YEAR
240    DATA 31,60,91,121,152,182,213,244,274,305,335
250    DATA 31,59,90,120,151,181,212,243,273,304,334
260    Y=0 @ FOR I=1 TO MONTH-1 @ READ Y@ NEXT I @ JDATE=DAY+Y
270    PRINT "JULIAN DATE: ",JDATE
280 CLEAR @ D$=" LATITUDE (degrees,minutes) example: 37deg 14min = 37,14 " @ DISP HGL?$ (D$,1) @ INPUT LAT1,MINUTES1
290 IF LAT1>60 OR MINUTES1>60 THEN BEEP @ CLEAR @ GOTO 280
300 LAT=LAT1+MINUTES1/60
310 DISP @ D$=" DIRECTION FROM EQUATOR (N/S) " @ DISP HGL?$ (D$,1) @ INPUT H$@ IF H$[1,1]="S" THEN LAT=-LAT
320 PRINT USING 330 ; LAT1,MINUTES1,H$
330 IMAGE "LATITUDE:    ",DDD," Degrees",X,DD," Minutes",X,A
340 IMAGE "LONGITUDE:   ",DDD," Degrees",X,DD," Minutes",X,A
350 CLEAR @ D$=" LONGITUDE (degrees,minutes) example: 34deg 14min = 37,14 " @ DISP HGL?$ (D$,1) @ INPUT LONG1,MINUTES2
360 IF LONG1>180 OR MINUTES2>60 THEN BEEP @ CLEAR @ GOTO 350
370 LONG=LONG1+MINUTES2/60
380 DISP @ D$=" DIRECTION FROM GREENWICH (E/W) " @ DISP HGL?$ (D$,1) @ INPUT J$@ IF J$[1,1]="W" THEN LONG=-LONG
390 PRINT USING 340 , LONG1,MINUTES2,J$
400 CLEAR @ D$=" MOON RISE AND SET TIMES BEING COMPUTED " @ AWRITE 9,15,HGL?$ (D$,1)
410 T0=(12-LONG/15)/24 @ T1=T0
420 D5=347.81 @ A=4 ! VALUE TO BE ANNUALLY UPDATED
430 GOSUB 630
440 Z3=Z @ Z4=Z1
450 H0=Z3+LONG
460 GOTO 490
470 GOSUB 630
480 IF D5<0 THEN D5=D5+360/ABS (X8)
490 X2=(.00233-SIN (LAT)*SIN (Z1))/(COS (LAT)*COS (Z1))
500 IF C=2 THEN GOTO 530
510 H1=360-ACS (X2)
520 GOTO 540
530 H1=ACS (X2)
540 X8=(H1-H0)/D5
550 IF X8>.5 THEN X8=X8-360/D5
560 IF X8<-.5 THEN X8=X8+360/D5
570 T2=T0+X8
580 IF ABS (T2-T1)<.01 THEN GOTO 1010
590 Q2=Q2+1
600 T1=T2
610 IF Q2>12 THEN Q3=1 @ GOTO 1010
620 GOTO 470
630 ! SUBROUTINE - CHG AND DEC [illegible]
```

```
660       W=N*8+1
670       P=MONTH*4-3+N
680    ! CHOOSE PROPER DATA TO USE IN SERIES EXPANSION CALCULATION
690    IF P)12 THEN 710
700    RESTORE 1800 @ GOTO 760
710    IF P)24 THEN 730
720    RESTORE 2070 @ GOTO 760
730    IF P)36 THEN 750
740    RESTORE 2340 @ GOTO 760
750    RESTORE 2610
760       FOR LOOP=1 TO P MOD 12
770       READ A8,B8,C8,D8,E8,F8,G8,H8,I8,J8,K8,L8,M8,N8,O8,P8
780       NEXT LOOP
790       RESTORE 1470
800    ! SERIES EXPANSIONS (7TH ORDER) FOR GHA AND DEC
810    X=(DAY+T1-W)/A-1
820       B7=X*H8
830       B6=X*(G8+B7)
840       B5=X*(F8+B6)
850       B4=X*(E8+B5)
860       B3=X*(D8+B4)
870       B2=X*(C8+B3)
880       B1=X*(B8+B2)
890       ! MOD = REMAINDER AFTER DIVISION BY 360
900    Z=(A8+B1) MOD 360 ! MOON'S GHA AT TIME T0
910       B7=X*P8
920       B6=X*(O8+B7)
930       B5=X*(N8+B6)
940       B4=X*(M8+B5)
950       B3=X*(L8+B4)
960       B2=X*(K8+B3)
970       B1=X*(J8+B2)
980    Z1=(I8+B1) MOD 360 ! MOON'S DECLINATION AT TIME T0
990 RETURN
1000  IF C=2 THEN GOTO 1060
1010  IF C=1 OR C=2 THEN GOSUB 1060
1020 GOTO 1030
1030 C=2 @ GOTO 410
1040 PAUSE
1050 END
1060 DAYS=T2+JDATE
1070 HOURS=(DAYS-IP (DAYS))*24
1080 MINUTES=INT ((HOURS-IP (HOURS))*60)
1090 IF C=1 THEN A$="MOONRISE"
1100 IF C=2 THEN A$="MOONSET" @ GOTO 1150
1110 PRINT @ PRINT @ SET I/O 7,16,1 @ PRINT " &18D" ! LINES 4051 THRU 4100 PRINTS RESULTS
1120 CLEAR @ D$=" MOON RISE AND SET TIMES BEING PRINTED " @ AWRITE 9,15,HGL?$ (D$,1)
1130 PRINT @ PRINT TAB (12);"JULIAN DATE";TAB (27),"ZULU TIME"
1140 PRINT TAB (12),"__________";TAB (27);"_________" @ SET I/O 7,16,2 @ PRINT @ PRINT " &16D"
1150 IF Q3=0 THEN 1160 ELSE 1180
1160 PRINT USING 1170 ; A$,IP (DAYS),IP (HOURS),IP (MINUTES)
1170 IMAGE 15A,DDD,10X,DD,":",ZZ
1180 IF Q3)0 THEN PRINT A$;TAB (12);"NONE-CHECK PREVIOUS AND FOLLOWING DAYS"
1190 IF C=2 THEN GOTO 1220
1200 Q3=0 @ Q2=0
1210 RETURN
1220 PRINT " &k2S" @ PRINT "TIMES ARE ACCURATE TO + OR - 5 MINUTES" @ PRINT @ PRINT " &k0S"
1230 CLEAR @ D$=" DO YOU WANT MOONRISE AND MOONSET TIMES FOR OTHER DATES OR LOCATIONS (Y/N) " @ DISP HGL?$ (D$,1) @ INPUT U6$
1240 IF U6$[1,1]="Y" THEN CLEAR @ GOTO 140
1250 PRINT " &k0S"
1260 DIM X$[75],Q(5)
1270 CLEAR @ D$=" PHASE DATES BEING PRINTED " @ AWRITE 9,20,HGL?$ (D$,1)
1280 IF JDATE)= 1 AND JDATE(= 31 THEN G$="JANUARY" @ RESTORE 1470
1290 IF JDATE)= 32 AND JDATE(= 59 THEN G$="FEBRUARY" @ RESTORE 1470
```

```
1320 IF JDATE)= 121 AND JDATE(= 151 THEN G$="MAY" @ RESTORE 1510
1330 IF JDATE)= 152 AND JDATE(= 181 THEN G$="JUNE" @ RESTORE 1520
1340 IF JDATE)= 182 AND JDATE(= 212 THEN G$="JULY" @ RESTORE 1530
1350 IF JDATE)= 213 AND JDATE(= 243 THEN G$="AUGUST" @ RESTORE 1540
1360 IF JDATE)= 244 AND JDATE(= 273 THEN G$="SEPTEMBER" @ RESTORE 1550
1370 IF JDATE)= 274 AND JDATE(= 304 THEN G$="OCTOBER" @ RESTORE 1560
1380 IF JDATE)= 305 AND JDATE(= 334 THEN G$="NOVEMBER" @ RESTORE 1570
1390 IF JDATE)= 335 AND JDATE(= 365 THEN G$="DECEMBER" @ RESTORE 1580
1400 ! LINES 6000 - 6110 GIVE PHASE/DATE INFO IN THE FORM W,Q(1),Q(2),Q(3),Q(4),Q(5)
1410 ! W SPECIFIES WHICH PHASE OCCURS FIRST IN THE MONTH.  IF W)=10, THEN FIVE PHASES OCCUR.
1420 ! W=1 OR 10, WANING CRESCENT      W=2 OR 20; NEW MOON
1430 ! W=3 OR 30, WAXING CRESCENT      W=4 OR 40, FULL MOON
1440 ! THE Q-VALUES ARE THE DAYS OF THE MONTH ON WHICH THE PHASES OCCUR (UPDATED ANNUALLY).
1450 !
1460 ! THE FOLLOWING 12 DATA STATEMENTS CONTAIN THE ABOVE INFORMATION FOR THE TWELVE MONTHS, ONE STATEMENT PER MONTH:
1470    DATA 2,3,11,18,25
1480    DATA 2,1,10,17,23
1490    DATA 2,2,10,17,24
1500    DATA 2,1,9,15,23
1510    DATA 2,1,8,15,22
1520    DATA 3,6,13,21,29
1530    DATA 3,5,13,21,28
1540    DATA 3,4,11,19,26
1550    DATA 3,2,10,18,25
1560    DATA 30,1,9,17,24,31
1570    DATA 4,8,16,22,30
1580    DATA 4,8,15,22,30
1590 READ W
1600 IF W)= 10 THEN 1620
1610 READ Q(1),Q(2),Q(3),Q(4)@ GOTO 1630
1620 READ Q(1),Q(2),Q(3),Q(4),Q(5)
1630 PRINT "THE APPROXIMATE DATES OF MOON PHASES FOR ",G$," ARE:" @ PRINT "PHASE/DATE"
1640 IF W=1 OR W=10 THEN X$="WANING CRESCENTNEW MOON       WAXING CRESCENTFULL MOON       "
1650 IF W=2 OR W=20 THEN X$="NEW MOON       WAXING CRESCENTFULL MOON       WANING CRESCENT"
1660 IF W=3 OR W=30 THEN X$="WAXING CRESCENTFULL MOON       WANING CRESCENTNEW MOON       "
1670 IF W=4 OR W=40 THEN X$="FULL MOON       WANING CRESCENTNEW MOON       WAXING CRESCENT"
1680 IF W=10 THEN X$=X$&"WANING CRESCENT"
1690 IF W=20 THEN X$=X$&"NEW MOON       "
1700 IF W=30 THEN X$=X$&"WAXING CRESCENT"
1710 IF W=40 THEN X$=X$&"FULL MOON       "
1720 IF W(10 THEN 1740
1730 FOR I=1 TO 5 @ PRINT X$[I*15-14,I*15];"/",Q(I) @ NEXT I @ GOTO 1750
1740 FOR I=1 TO 4 @ PRINT X$[I*15-14,I*15];"/";Q(I) @ NEXT I
1750 CHAIN "MENU.DISC1"
1760 ! DATA FOR FIRST 12 PERIODS OF SIXTEEN TERMS:
1770 !       MOON GHA FOR TERMS 0 - 7.  LINES 1,3,5, AND 7
1780 !       MOON DEC FOR TERMS 0 - 7.  LINES 2,4,6, AND 8
1790 ! DATA FOR JANUARY.
1800 DATA 1956.6967,1392.2494,4.7149,1.0619,-1.3222,-.041,.1996 -.0365
1810 DATA -23.0755,8.7563,8.1968,-1.9327,-.3374,.3147,-.0184,-.0396
1820 DATA 1512.9702,1395.8771,-6.3791,-2.7110,.4453,.5095,.1491,-.0126
1830 DATA 12.013,19.3377,-2.9381,-2.6935,-.7391,-.1018,.1222,.0513
1840 DATA 1401.7515,1386.0607,7.0169,-.8042,-2.1005,.9104,.2125,-.2131
1850 DATA 14.7271,-21.1826,-6.7235,4.9178,-.0434,-.7043,.1062,.0636
1860 DATA 1664.3651,1389.577,-1.1272,2.0618,.8783,-.4557,-.1409,.0379
1870 DATA -24.0859,-6.7991,9.8414,.6801,-.7801,-.1219,.0444,.0213
1880 ! DATA FOR FEBRUARY:
1890 DATA 1503.2818,1400.0894,2.3508,-1.9301,-.3858,.102,-.0344,.0177
1900 DATA -9.3321,18.9236,2.7345,-1.7525,.142,-.0355,-.0326,.0029
1910 DATA 1494.3531,1382.9105,-6.9839,3.3789,2.4793,-.4741,-.5305,-.0039
1920 DATA 24.7242,5.5618,-11.9805,-3.141,1.1162,.8981,-.029,-.1379
1930 DATA 1743.5085,1391.4612,.61,-2.2693,.5491,.3466,-.08,-.0014
1940 DATA -7.1409,-23.6776,4.4485,3.1948,-.7645,.0781,.0332,-.0783
1950 DATA 1644.8922,1393.9756,4.2021,-.412,-.9961,.852,.1217,-.134
```

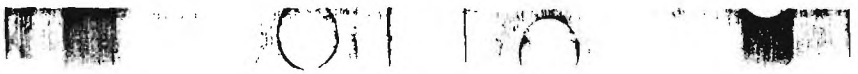
```
1980 DATA 1593.0284,1400.9025,  .428,  2.1774,  .1587,.0848,.0188,.0112
1990 DATA -.5209,20.5402,.4576,-1.7649,  .048,-.0877,-.0124,.0072
2000 DATA 1497.7162,1381.5805,-1.2897,4.6442,-.0689,-1.3327,.0084,.234
2010 DATA 25.0777,-5.121,-13.3505,-.2399,1.9232,.2531,-.2229,-.036
2020 DATA 1750.8485,1389.1137,-1.572,-.4229,1.1711,.1417,-.1366,-.0216
2030 DATA -16.0086,-19.4531,8.3071,2.5415,-.9222,-.0616,.0189,.0049
2040 DATA 1652.7643,1397.8724,3.8831,-1.364,-.5177,.2235,-.018,-.0292
2050 DATA -16.0829,16.3826,4.9265,-1.0491,.0788,-.019,-.0383,.0227
2060 ! DATA FOR APRIL:
2070 DATA 1580.8114,1394.0368,-6.1669,-1.2152,.9618,.4734,-.0222,-.0827
2080 DATA 17.8407,16.3132,-5.7970,-2.7625,-.2204,.1394,.0891,.0101
2090 DATA 1474.0227,1388.4301,3.1077,-1.529,-1.0297,.5295,.0683,-.0578
2100 DATA 10.9676,-23.7277,-5.6196,4.2293,.471,-.2849,.0133,.0041
2110 DATA 1730.4134,1387.148,3.0262,2.8268,-.6343,-.6551,.1302,.0737
2120 DATA -25.8676,-.8498,11.0932,-1.1508,-.9491,.3261,.0612,-.0407
2130 DATA 1642.1874,1400.4176,-1.6868,-2.2857,.0006,.0481,.047,.0397
2140 DATA 1.9147,21.0725,.0251,-1.903,-.2588,-.1089,.0219,.02
2150 ! DATA FOR MAY:
2160 DATA 1572.6964,1385.2009,-3.963,3.4604,1.2123,-.7677,-.27,.0921
2170 DATA 25.5536,4.325,-11.5661,-1.7789,1.2048,.3956,-.1147,-.0641
2180 DATA 1468.1854,1391.1379,-1.8198,-2.2545,.6303,.4641,-.0107,-.0245
2190 DATA -5.6221,-24.9319,2.575,4.014,-.0204,-.1469,-.0657,-.0235
2200 DATA 1724.1429,1392.9074,6.284,-.3137,-1.3945,.3316,.1312,-.0704
2210 DATA -22.0919,11.7944,7.4617,-2.4286,.1157,.2405,-.1031,-.0026
2220 DATA 1641.1095,1394.8413,-6.4668,-1.712,.957,.5667,.0309,-.0839
2230 DATA 15.4139,10.3217,-4.6046,-2.9777,-.4553,.1163,.1227,.0297
2240 ! DATA FOR JUNE:
2250 DATA 1547.4293,1387.2098,5.3086,.0616,-2.0493,.4293,.2691,-.1264
2260 DATA 18.9816,-17.4697,-8.8187,3.3523,.4594,-.4374,.0864,.0448
2270 DATA 1448.3706,1386.1382,-2.6357,2.7725,1.5622,-.5559,-.2966,.043
2280 DATA -23.266,-11.135,11.0061,1.7642,-1.2924,-.323,.12,.0635
2290 DATA 1712.2374,1400.7844,1.7076,-2.5578,-.1633,.0884,-.045,.0233
2300 DATA -6.6623,20.132,1.9087,-1.5841,.1308,-.1268,-.0221,-.003
2310 DATA 1620.9773,1382.3238,-3.5903,5.1006,1.247,-1.4171,-.3094,.2436
2320 DATA 25.8664,1.9682,-12.9292,-1.6094,1.8573,.4958,-.2486,-.0818
2330 ! DATA FOR JULY:
2340 DATA 1539.9118,1392.5226,2.3683,-2.9066,-.2307,.4225,-.0713,.03
2350 DATA 3.6871,-24.7146,-1.0377,3.5689,-.3119,.0299,.0682,-.03
2360 DATA 1439.2353,1387.7747,4.0017,3.0247,-1.2247,-.6589,.2566,.081
2370 DATA -25.5211,4.2716,10.5729,-1.7020,-.9885,.4038,.0924,-.0686
2380 DATA 1713.1439,1400.0512,-3.2676,-2.745,.0114,.1602,.009,.0202
2390 DATA 7.2656,20.331,-1.5186,-1.821,-.3371,-.1435,.0103,.0021
2400 DATA 1609.9212,1382.898,4.8869,2.7315,-2.5614,-.3046,.5194,-.024
2410 DATA 21.4166,-15.3099,-11.4562,3.4808,1.5131,-.7695,-.1291,.1459
2420 ! DATA FOR AUGUST:
2430 DATA 1521.7581,1390.0136,-3.1559,-.4979,1.4662,.2542,-.156,-.0598
2440 DATA -17.945,-17.8421,8.2103,2.4335,-.4986,-.131,-.0506,.0044
2450 DATA 1422.4886,1398.1519,4.5249,-1.6817,-.6094,.3282,-.0329,-.0497
2460 DATA -14.1715,18.0313,4.0206,-2.2021,.2746,.0386,-.0702,.012
2470 DATA 1699.314,1389.5166,-7.9054,.1184,1.9809,.5594,-.2556,-.1807
2480 DATA 23.0191,11.3155,-8.628,-3.2442,-.082,.4967,.1532,-.0478
2490 DATA 1589.387,1390.0255,1.9681,-2.5713,-.1632,.5191,-.0639,.0068
2500 DATA .9966,-26.4913,.2751,4.5876,-.5062,-.2009,-.0985,-.0132
2510 ! DATA FOR SEPTEMBER:
2520 DATA 1499.4632,1307.86,3.1033,2.7493,-.671,-.6492,.1419,.0874
2530 DATA -26.1379,1.863,10.8802,-1.2425,-.0802,.229,.0679,-.0254
2540 DATA 1772.6156,1401.1619,-1.5756,-2.2934,-.0367,.0834,.0441,.0103
2550 DATA 4.7582,20.7160,-1.1496,-1.7675,-.0704,  .1108,-.0048,.0136
2560 DATA 1676.0889,1383.2314,.0292,3.5993,-1.1083,-.9428,.2332,.1447
2570 DATA 24.34[illegible],10.183,-12.7724,.9527,1.7519,  .8319,-.1848,.0247
2580 DATA 1570.462,1386.3477,-2.48,1.005,1.6181,-.0566,-.2634,-.0336
2590 DATA -20.1704,17.0188,10.4781,2.395,-1.2311,-.1715,.0662,.0262
2600 ! DATA FOR OCTOBER.
```

```
2640 DATA 18.032,16.5073,-5.5514,-2.4786,-.2008,.116,.0737,-.0066
2650 DATA 1667.4704,1388.7175,2.5867,-1.3861,-1.2037,.4924,.1126,-.0845
2660 DATA 13.1067,-23.1856,-6.9943,3.7124,.7437,-.1571,-.0051,-.0111
2670 DATA 1561.2979,1384.1831,3.2936,3.9844,-.7886,-1.0501,.1816,.1618
2680 DATA -26.5635,-1.5612,12.5639,-1.3169,-1.4108,.4404,.1303,-.0602
2690 ! DATA FOR NOVEMBER
2700 DATA 1481.0168,1401.527,.685,-2.4076,.0302,.0506,-.0196,.0195
2710 DATA -3.2735,21.0512,1.0065,-1.6591,-.0259,-.1506,.0081,.0210
2720 DATA 1751.1928,1385.827,-1.7277,3.7418,.2249,-1.012,-.0672,.1642
2730 DATA 26.6394,-.456,-11.8156,-.6973,1.2636,.147,-.136,-.0212
2740 DATA 1649.4163,1389.353,-4.262,-1.9732,1.2011,.6879,-.0315,-.0883
2750 DATA -10.0935,-24.8188,4.4746,4.562,.0652,-.3404,-.1244,-.0053
2760 DATA 1541.8447,1394.6763,7.0573,-1.7195,-1.101,.6231,-.0114,-.0927
2770 DATA -19.1307,15.9411,5.0256,-2.7344,.4998,.101,-.1422,.0256
2780 ! DATA FOR DECEMBER
2790 DATA 1481.6577,1399.2582,-4.2324,-2.3219,.3609,.2527,.0877,.003
2800 DATA 10.8023,19.9927,-2.6535,-2.1368,-.4263,-.0802,.0697,.0302
2810 DATA 1740.7656,1387.9731,4.6735,.5151,-2.0277,.264,.2908,-.1162
2820 DATA 20.6523,-16.1923,-9.4108,2.8479,.6082,-.4284,.0436,.0679
2830 DATA 1641.5314,1384.1566,-4.4209,2.9232,2.3458,-.4248,-.4711,-.0073
2840 DATA -23.1024,-13.276,11.5975,2.8067,-1.412,-.6415,.1165,.1218
2850 DATA 1551.2149,1398.8825,4.7572,-2.5992,-.3171,.29,-.1149,.0133
2860 DATA -11.63,19.7582,2.8291,-2.1343,.504,-.1033,-.0728,.023
```

```
4260 ! SUNRISE/SUNSET/TWILIGHT TIMES (VALID FOR APPROX 60degN TO 60degS LAT)
4270 !
4280 DIM W$[21]
4281 GCLEAR @ GRAPHALL @ MSCALE 0,0
4282 CSIZE 9
4283 MOVE 17,55 @ LABEL "SUNRISE/SUNSET/TWILIGHT TIMES"
4284 CSIZE 5 @ MOVE 35,49 @ LABEL "(VALID FOR LATITUDES 60degN TO 60degS)"
4285 MOVE 45,9 @ LABEL "PRESS 'CONT' WHEN READY..." @ PAUSE
4286 GCLEAR
4287 PRINT " &l1V";" &k0S" @ FOR I=1 TO 32 @ PRINT "SUN-";@ NEXT I @ PRINT "SUN";@ PRINT " &l5V"
4290 PRINT " &k1S";" &dD";"SUNRISE/SUNSET/TWILIGHT TIMES";" &dA" @ PRINT " &k0S" @ ALPHA
4295 PRINT "(VALID FOR LATITUDES 60degN TO 60degS)";" &k3S" @ PRINT @ PRINT
4300 DEG @ CLEAR @ DISP "ENTER THE MONTH,DAY,YEAR (e.g. 10,28,1982)." @ INPUT MONTH,DAY,YEAR
4310 PRINT USING 4320 ; MONTH,DAY,YEAR
4320 IMAGE "DATE: ",DD,"/",ZZ,"/",DDDD
4330 IF YEAR/4=INT (YEAR/4) THEN RESTORE 4350 ELSE RESTORE 4360 ! DETERMINES LEAP YEAR
4340 ! LINE 4050 = LEAP YEAR (366 DAYS)   LINE 4060 = 365-DAY YEAR
4350 DATA 31,60,91,121,152,182,213,244,274,305,335
4360 DATA 31,59,90,120,151,181,212,243,273,304,334
4370 Y=0 @ FOR I=1 TO MONTH-1 @ READ Y@ NEXT I @ JDATE=DAY+Y
4380 CLEAR @ DISP "LATITUDE (degrees, minutes)     e.g. 37deg 14min N = 37,14" @ INPUT LAT1,MINUTES1
4390 LAT=LAT1+MINUTES1/60
4400 DISP @ DISP "DIRECTION FROM EQUATOR (N/S)";@ INPUT H$@ IF H$[1,1]="S" THEN LAT=-LAT
4410 PRINT USING 4420 ; LAT1,MINUTES1,H$
4420 IMAGE "LATITUDE:  ",DDD," Degrees",X,DD," Minutes",X,A
4430 IMAGE "LONGITUDE:  ",DDD," Degrees",X,DD," Minutes",X,A
4440 CLEAR @ DISP "LONGITUDE (degrees, minutes)    e.g. 34deg 14min E = 34,14" @ INPUT LONG1,MINUTES1
4450 LONG=LONG1+MINUTES1/60
4460 DISP @ DISP "DIRECTION FROM GREENWICH (E/W)";@ INPUT J$@ IF J$[1,1]="E" THEN LONG=-LONG @ CLEAR
4470 PRINT USING 4430 , LONG1,MINUTES1,J$
4480 PRINT @ PRINT @ PRINT " &l8D" @ SET I/O 7,16,1
4490 PRINT USING 4500
4500 IMAGE 26X,"RISING",6X,"SETTING",10X,"DURATION"
4510 PRINT "                          ________  ________  ________________"
4520 SET I/O 7,16,2 @ PRINT @ PRINT " &l6D"
4530 T=JDATE+(6+LONG/15)/24 ! APPROXIMATE TIME OF PHENOMENON IN DAYS SINCE 0 JAN, 0 hrs UT
4540 M=.9856*T-3.289 ! SUN'S MEAN ANOMALY
4550 ! COMPUTE SUN'S TRUE LONGITUDE (L)
4560    L=(M+1.916*SIN (M)+.02*SIN (2*M)+282.634) MOD 360
4570    IF L=180 THEN L=179.99999999
4580    IF L=270 THEN L=269.99999999
4590 ! CALCULATE SUN'S RIGHT ASCENSION (RA)
4600    RA=ATN (.91746*TAN (L))
4610       ! PLACE RA IN SAME QUADRANT AS L
4620       IF L)90 AND L(180 THEN RA=90-ABS (RA)+90
4630       IF L)= 180 AND L(270 THEN RA=ABS (RA)+180
4640       IF L)270 THEN RA=90-ABS (RA)+270
4650    RA=RA/15 ! CONVERT HOURS TO DEGREES
4660 D=ASN (.39782*SIN (L)) ! SUN'S DECLINATION
4670 FOR I=1 TO 4
4680    ! DETERMINE COS OF ZENITH DIST FOR EACH PHENOMENON (Z)
4690       IF I=1 THEN Z=-.0145439 @ W$="SUNRISE AND SUNSET"
4700       IF I=2 THEN Z=-.1045285 @ W$="CIVIL TWILIGHT"
4710       IF I=3 THEN Z=-.2079117 @ W$="NAUTICAL TWILIGHT"
4720       IF I=4 THEN Z=-.309017 @ W$="ASTRONOMICAL TWILIGHT"
4730    H=(Z-SIN (D)*SIN (LAT))/(COS (D)*COS (LAT)) ! SUN'S LOCAL HOUR ANGLE
4740    IF H(= 1 AND H)= -1 THEN H=ACS (H) @ GOTO 4790
4750    ! PRINTOUT FOR ABSENCE OF PHENOMENON (/COS H/)0)
4760       PRINT USING 4770 , W$
4770       IMAGE 29A,"*",12X,"*",16X,"*"
```

```
4805 IF T1<T2 THEN GOTO 4810 ! T1=RISETIME  T2=SETTIME
4806 IF T1>T2 THEN GOTO 4815
4810    SPAN=ABS (T1-T2) @ GOTO 4820 ! TIME SPAN COMPUTED
4815    SPAN=ABS (T2+24-T1) ! TIME SPAN COMPUTED
4820    ! PRINTOUT FOR TIME OF PHENOMENON
4830       H1=ABS (T1\1) @ H2=ABS (T2\1) @ S1=SPAN\1
4840       M1=ABS (ABS (T1) MOD H1*60) @ M2=ABS (ABS (T2) MOD H2*60) @ S2=ABS (SPAN MOD S1*60)
4850       IF T1<0 THEN B1$="-" ELSE B1$=" "
4860       IF T2<0 THEN B2$="-" ELSE B2$=" "
4865 H1=H1 MOD 24
4866 H2=H2 MOD 24
4870       PRINT USING 4880 ; W$,B1$,H1,M1,B2$,H2,M2,S1,S2
4880       IMAGE 23A,A,ZZ,":",ZZ," ZULU",2X,A,ZZ,":",ZZ," ZULU",3X,DD," HOURS ",DD," MINUTES"
4890 NEXT I
4900 PRINT " &k2S" @ PRINT "VALUES ARE ACCURATE TO WITHIN + OR - 5 MINUTES."
4910 PRINT "* - TWILIGHT LASTS ALL NIGHT"
4911 CLEAR @ DISP "WOULD YOU LIKE TO WORK ANOTHER PROBLEM (Y/N)" @ INPUT I$
4912 IF I$[1,1]="Y" THEN PRINT " &16V" @ GOTO 4290
4915 PRINT " &k0S";" &12V" @ FOR I=1 TO 32 @ PRINT "SUN-",@ NEXT I @ PRINT "SUN"
4920 CLEAR @ DISP "PLEASE WAIT" @ CHAIN "MENU.DISC1"
```

```
10 ! OPTS458.DISC1
20 ! CONTAINS MENU OPTIONS 4=DENSITY ALTITUDE, 5=ATMOSPHERIC STABILITY, and 8=SUNRISE/SUNSET/TWILIGHT TIMES
30 COM B(20),C,X,W$[60],T,W,G,E,V,S,P$[1],R ,INTEGER D(192)
40 PRINTER IS 506,132
45 DIM D$[80],D1$[80]
50 ON X GOTO 60,2930,4250
60 !
70 ! DENSITY ALTITUDE
80 !
81 GCLEAR @ GRAPH @ MSCALE 0,0
82 CSIZE 9
84 MOVE 10,55 @ LABEL "DENSITY ALTITUDE PROGRAM"
85 CSIZE 5 @ MOVE 33,9 @ LABEL "PRESS 'CONT' WHEN READY..."
86 PAUSE
87 GCLEAR
88 PRINT " &11V" @ FOR I=1 TO 11 @ PRINT "DENSITYALT-";@ NEXT I @ PRINT "DENSITYALT"," &15V"
90 ON ERROR GOTO 90 @ CLEAR @ DISP "STATION ELEVATION (feet)";@ INPUT ELEV
100 ON ERROR GOTO 100 @ CLEAR @ DISP "TEMPERATURE (degrees F)";@ INPUT TEMP
110 DISP @ DISP "DEW POINT (degrees F)";@ INPUT FDEWPT
120 CDEWPT=5*(FDEWPT-32)/9
130 ON ERROR GOTO 130 @ CLEAR @ DISP "ENTER THE UNCORRECTED STATION PRESSURE (inches of mercury).  ENTER A '0' IF ONLYTHE PRESSURE A
LTITUDE IS KNOWN."
140 DISP @ INPUT BP ! BAROMETRIC PRESSURE
150 IF BP=0 THEN GOSUB 570 ELSE GOSUB 710
160 BP=INT (BP*100)/100 @ PA=INT (PA)
170 ICEPT=273.16
180 STDATM=1013.246
190 STEAMPT=373.16
200 IF CDEWPT<= 0 THEN GOTO 330
210 CDEWPT=CDEWPT+ICEPT
220 D=-(7.90298*(STEAMPT/CDEWPT-1))
230 E=5.02808*LGT (STEAMPT/CDEWPT)
240 R=1-CDEWPT/STEAMPT
250 F=1.3816*10^-7*(10^(11.344*R)-1)
260 S=STEAMPT/CDEWPT-1
270 G=8.1328*10^-3*(10^(-(3.49149*S))-1)
280 H=LGT (STDATM)
290 I=D+E-F+G+H
300 Z=10^I
310 SVP=Z*.02953
320 GOTO 410
330 D2=CDEWPT+ICEPT
340 D=(-9.09718)*(ICEPT/D2-1)
350 E=(-3.56654)*LGT (ICEPT/D2)
360 F=.876793*(1-D2/ICEPT)
370 G=LGT (6.1071)
380 I=D+E+F+G
390 Z=10^I
400 SVP=Z*.02953
410 CLEAR
420 R=.622*SVP/(BP-SVP) ! MIXING RATIO CALCULATION
430 I=(TEMP+459.69)*((1+1.61*R)/(1+R)) ! VIRTUAL TEMPERATURE CALCULATION
440 DA=INT ((1-(17.326*BP/I)^.235)*145366) ! DENSITY ALTITUDE FORMULA:SMITSNIN MET TABLES
450 PRINT " &k1S" @ PRINT " &13D"
460 SET I/O 7,16,1
470 PRINT "DENSITY ALTITUDE CALCULATIONS:" @ PRINT "_____"
480 SET I/O 7,16,2
490 PRINT " &k0S" @ PRINT " &16D"
500 PRINT "STATION ELEVATION = ";ELEV;" feet"
```

```
510 PRINT "TEMPERATURE       = ";TEMP;" degrees fahrenheit"
520 PRINT "DEW POINT         = ";FDEWPT;" degrees fahrenheit"
530 PRINT "PRESSURE          = ";BP;" inches of mercury"
540 PRINT "PRESSURE ALTITUDE = ";PA;" feet"
550 PRINT " &14D" @ PRINT " &k3S"
560 PRINT @ PRINT "YOUR DENSITY ALTITUDE IS ";DA;" feet" @ GOTO 800
570  !
580  ! DETERMINES PRESSURE ALT FROM BAROMETRIC PRESSURE
590  !
600  PA=BP*33.8639 ! CONVERTS INCHES HG TO MILLIBARS
610  IF PA>= 1013.25 THEN PA=26859.7345224-26.5185*PA @ GOTO 640
620  IF PA<705.09 THEN PA=50125.7923315-79.0246642817*PA+.030808325969*PA^2 @ GOTO 640
630  PA=42648.3525204-57.4057011468*PA+1.51260276686E-2*PA^2
640     IF PA<0 THEN C=7.13966666655+.0264698*PA+.0000135663*PA^2 @ GOTO 690
650     IF PA<5000 THEN C=-11.4763604151+1.40661336336E-2*PA-2.36456659055E-6*PA^2 @ GOTO 690
660     IF PA<9000 THEN C=-1175.00071+.73075*PA-.000165866*PA^2+.000000016224*PA^3-5.78465E-13*PA^4 @ GOTO 690
670     IF PA>9350 AND PA<9700 THEN C=118740.544813-38.10748*PA+.0040738*PA^2-.000000145063*PA^3 @ GOTO 690
680     IF PA>= 9700 THEN C=-1273.73891+.2686*PA-.0000183266*PA^2+4.05324E-10*PA^3
690  PA=PA+C
700  RETURN
710  !
720  ! DETERMINES BAROMETRIC PRESSURE FROM PRESSURE ALT
730  !
740  CLEAR @ DISP "PRESSURE ALTITUDE (feet)" @ INPUT PA
750  BP=1013.08968413-3.60310856232E-2*PA+4.38653185363E-7*PA^2
760  C=-.1403422+.000558757*PA-.000000091921*PA^2+3.46176E-12*PA^3
770  BP=BP-C
780  BP=BP/33.8639 ! CONVERTS MILLIBARS TO INCHES HG
790 RETURN
800 !
810 ! HELICOPTER LOADING CAPACITIES
820 !
830 CLEAR @ DISP "DO YOU WANT LOAD CAPACITIES FOR SELECTED HELICOPTERS AT THE COMPUTED DENSITY    ALTITUDE (Y/N)" @ INPUT I$
840 IF I$[1,1]="N" THEN PRINT " &11V" @ CLEAR @ DISP "PLEASE WAIT" @ CHAIN "MENU.DISC1"
850 DIM HELTYP$[50]
860 TEMP=(TEMP-32)*5/9 ! TEMP NOW IN DEG C
870 D$=" ENTER THE NUMBER OF THE HELICOPTER (1-14) " @ D1$="" @ FOR I=1 TO LEN (D$) @ D1$=D1$&CHR$ (NUM (D$[I,I])+128) @ NEXT I
880 ON ERROR GOTO 880 @ CLEAR @ DISP TAB (19);D1$ @ DISP
890 DISP TAB (15);"   1.  AH-1G                          8.  EH-1H"
900 DISP TAB (15),"   2.  AH-1S                          9.  OH-6A"
910 DISP TAB (15);"   3.  CH-47A                        10.  OH-58A"
920 DISP TAB (15),"   4.  CH-47B                        11.  OH-58C"
930 DISP TAB (15),"   5.  CH-47C                        12.  UH-1C/M"
940 DISP TAB (15);"   6.  CH-54A                        13.  UH-1D/H"
950 DISP TAB (15);"   7.  CH-54B                        14.  UH-60A" @ DISP @ INPUT H
960 ON H GOSUB 1110,1230,1350,1470,1620,1770,1950,2620,2140,2260,2380,2500,2620,2750
970 PRINT " &k1S";" &15V" @ SET I/O 7,16,1
980 PRINT "HELICOPTER LOADING CAPABILITY" @ PRINT "_____________________________"
990 SET I/O 7,16,2 @ PRINT @ PRINT " &k0S";" &16D"
1000 TEMP=INT (TEMP*100)/100 @ PA=INT (PA) @ DA=INT (DA) @ WT=INT (WT/10)*10
1010 PRINT "HELICOPTER TYPE:  ";HELTYP$
1020 PRINT "OAT = ";TEMP;" degrees Centigrade,     Pressure Altitude = ";PA;
1030 PRINT " feet,     Density Altitude = ";DA;" feet" @ PRINT " &k3S" @ PRINT
1040 PRINT "APPROXIMATE MAXIMUM ALLOWABLE LOAD (CARGO, FUEL, AND PASSENGER MIX) = ";WT;" LBS."
1050 PRINT " &k2S";"(ALLOWABLE LOAD ASSUMES AIRCRAFT BASIC WEIGHT = ";CRAFT;" LBS.,  OIL = ";OIL;
1060 PRINT " LBS., AND CREW (";CREW;") = ";CREW*200;" LBS.)" @ PRINT " &k0S"
1070 CLEAR @ DISP "DO YOU WANT LOAD CAPACITIES FOR OTHER HELICOPTERS AT THE SAME DENSITY          ALTITUDE (Y/N)" @ INPUT B$
1080 IF B$[1,1]="Y" THEN 880
1085 PRINT " &15V" @ FOR I=1 TO 11 @ PRINT "DENSITYALT-";@ NEXT I @ PRINT "DENSITYALT"
1090 CLEAR @ DISP "PLEASE WAIT" @ CHAIN "MENU.DISC1"
```

```
1110 !
1120 ! AH-1G (TM 55-1520-221-10   DATE: 18 MARCH 1980)
1130   CRAFT=6067 @ OIL=24 @ CREW=1 @ HELTYP$="AH-1G (T53-L-13B)" @ MAXWT=9500 @ MAXTORQUE=50 ! PSI
1140      A=98872.516782-177.53792*TEMP+3.058016*TEMP^2+.1009999*TEMP^3-.00465033*TEMP^4
1150      B=-23549.2176-5.53587*TEMP-1.18097*TEMP^2-.0315638*TEMP^3+.00135519*TEMP^4
1160      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
1170   YSCALE=.2*TORQUE-5
1180      A=5539.57727-.00467087*DA-.000000847021*DA^2
1190      B=812.884-.014*DA
1200      GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
1210   WT=GROSSWT-(CRAFT+OIL+CREW*200)
1220   RETURN
1230 !
1240 ! AH-1S (TM 55-1520-234-10   DATE: 17 NOVEMBER 1976)
1250   CRAFT=6910 @ OIL=29 @ CREW=1 @ HELTYP$="AH-1S (T53-L-703)" @ MAXWT=10000 @ MAXTORQUE=56 ! PSIG
1260      A=85128.094-391.18*TEMP+6.933*TEMP^2-.0974*TEMP^3
1270      B=-18754.672+78.948*TEMP-2.1278*TEMP^2-.02847*TEMP^3+.00094623*TEMP^4
1280      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
1290   YSCALE=.2*TORQUE-5
1300      A=5587.655+.0004856*DA+.00000118276*DA^2-9.194E-11*DA^3
1310      B=797.3967-.01379*DA-.000006807*DA^2+2.86E-11*DA^3
1320      GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
1330   WT=GROSSWT-(CRAFT+OIL+CREW*200)
1340   RETURN
1350 ! CH-47A (TM 55-1520-209-10   DATE:  9 JANUARY 1979)
1360   CRAFT=17105 @ OIL=28 @ CREW=2 @ HELTYP$="CH-47A (T55-L-7/7B), TWO-ENGINE OPERATION" @ MAXWT=33000 @ MAXTORQUE=860 ! LB-FT
1370      A=180909.633-100.4284*TEMP-9.0486*TEMP^2+.2776*TEMP^3-.0026125*TEMP^4
1380      B=-26178.3089-9.0711*TEMP+1.25465*TEMP^2-.04209*TEMP^3+.00038448*TEMP^4
1390      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
1400   YSCALE=.01*TORQUE-6
1410      A=17865.134-.03225*DA+.0000066797*DA^2-8.496E-10*DA^3
1420      B=2029-.0399*DA-.000002127*DA^2+2.382E-10*DA^3
1430      GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
1440   WT=GROSSWT-(CRAFT+OIL+CREW*200)
1450   RETURN
1460 !
1470 ! CH-47B (TM 55-1520-227-10-1   DATE:  23 AUGUST 1978)
1480   CRAFT=19410 @ OIL=28 @ CREW=2 @ HELTYP$="CH47B (T55-L-7B), TWO-ENGINE OPERATION" @ MAXWT=40000 @ MAXTORQUE=860 ! LB-FT
1490      A=180909.633-100.4284*TEMP-9.0486*TEMP^2+.2776*TEMP^3-.0026125*TEMP^4
1500      B=-26178.3089-9.0711*TEMP+1.25465*TEMP^2-.04209*TEMP^3+.00038448*TEMP^4
1510      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
1520   YSCALE=.01*TORQUE-3
1530      IF TEMP)10 THEN A=0 ELSE A=-.016543+.00069857*TEMP+.0000964286*TEMP^2
1540      IF TEMP)10 THEN B=1 ELSE B=.98997+.000602381*TEMP+.0000142857*TEMP^2+.0000028333*TEMP^3
1550      Y=A+B*YSCALE
1560   A=18600.2389-.00461*DA-.0000048954*DA^2
1570   B=3938.1889-.07542786*DA+.0000011B474*DA^2
1580   GROSSWT=A+B*Y @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
1590   WT=GROSSWT-(CRAFT+OIL+CREW*200)
1600   RETURN
1610 !
1620 ! CH-47C (TM 55-1520-227-10-2   DATE:  23 AUGUST 1978)
1630   CRAFT=21791 @ OIL=28 @ CREW=2 @ HELTYP$="CH-47C (T55-L-7C), TWO-ENGINE OPERATION" @ MAXWT=40000 @ MAXTORQUE=990 ! LB-FT
1640      A=1012.8261-.039624*PA+.00000065447*PA^2
1650      B=-7.22367+.0003429*PA+7.6667E-9*PA^2
1660      TORQUE=A+B*TEMP @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
1670   YSCALE=.0078*TORQUE-3.12
1680      A=-.042277-.00041146*TEMP+.0000907749*TEMP^2-.0000013535*TEMP^3
1690      B=1.06055-.003278*TEMP+.000043958*TEMP^2
```

```
1700        Y=(YSCALE-A)/B
1710     A=22070.8728-.0336509*DA-.00000170867*DA^2+3.01605E-10*DA^3
1720     B=4997.193-.114*DA
1730     GROSSWT=A+B*Y @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
1740     WT=GROSSWT-(CRAFT+OIL+CREW*200)
1750     RETURN
1760 !
1770 ! CH-54A (TM 55-1520-217-10-1   DATE:  8 APRIL 1977)
1780     CRAFT=21895 @ OIL=15 @ CREW=2 @ HELTYP$="CH-54A (T73-P-1), TWO-ENGINE OPERATION" @ MAXWT=40000 @ MAXTORQUE=81.5 ! PERCENT
1790        IF TEMP(= -11 THEN 1820
1800        A=34787.769488-70.79067*TEMP+.7604*TEMP^2
1810        B=-296.62895-.2883*TEMP-.03579*TEMP^2 @ GOTO 1840
1820        A=35613.1689-1.22378*TEMP+.56865*TEMP^2
1830        B=-296.125-.166*TEMP-.007698*TEMP^2
1840        TORQUE=(PA-A)/B @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
1850     YSCALE=.05*TORQUE-1
1860        A=.98398+.0004568*TEMP-.0000029856*TEMP^2
1870        B=.935+.0012618*TEMP
1880        Y=A*YSCALE^B
1890     A=25925.7756-.167786*DA-.000061351*DA^2
1900     B=.4199-.00000075284*DA-5.4282E-11*DA^2
1910     GROSSWT=A*Y^B @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
1920     WT=GROSSWT-(CRAFT+OIL+CREW*200)
1930     RETURN
1940 !
1950 ! CH-54B (TM 55-1520-217-10-2   DATE:  15 APRIL 1977)
1960     CRAFT=22625 @ OIL=15 @ CREW=2 @ HELTYP$="CH-54B (T73-P-700), TWO-ENGINE OPERATION" @ MAXWT=47000 @ MAXTORQUE=82.5 ! PERCENT
1970        IF TEMP)= -14 THEN 2000
1980        A=94594.5894-3884.3305*TEMP-184.65234*TEMP^2-3.58638*TEMP^3-.02472*TEMP^4
1990        B=-21752.01913+473.1486*TEMP+16.41736*TEMP^2+.17233*TEMP^3 @ GOTO 2020
2000        A=121998.47092-95.3369*TEMP-7.31065*TEMP^2
2010        B=-26009.6487-30.9293*TEMP+.84051*TEMP^2+.102388*TEMP^3-.00172256*TEMP^4
2020        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2030     YSCALE=.05*TORQUE-1
2040        A=.9892857+.0004166667*TEMP-.00001382857*TEMP^2+.00000019733*TEMP^3
2050        B=.9479+.00171*TEMP-.000013371*TEMP^2
2060        Y=A*YSCALE^B
2070     A=16542.1559+.1092*DA-.00000203965*DA^2-.00000000067*DA^3
2080     B=14187.787-.3305*DA-.0000099711*DA^2+1.2315E-9*DA^3
2090     C=-1253.3072+.03978*DA+.0000010865*DA^2-2.325E-10*DA^3
2100     GROSSWT=A+B*Y+C*Y^2 @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2110     WT=GROSSWT-(CRAFT+OIL+CREW*200)
2120     RETURN
2130 !
2140 ! OH-6A (TM 55-1520-214-10   DATE:  17 DECEMBER 1976)
2150     CRAFT=1050 @ OIL=6 @ CREW=1 @ HELTYP$="OH-6A (T63-A-5A/-700)" @ MAXWT=2550 @ MAXTORQUE=80.3 ! PSIG
2160        A=10 775.08329-248.8135*TEMP+17.0171*TEMP^2-1.09376*TEMP^3+.0163997*TEMP^4
2170        B=-2.664.3551+17.37122*TEMP-4.5003*TEMP^2+.28398*TEMP^3-.0043237*TEMP^4
2180        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2190     YSCALE=.2*TORQUE-8
2200        A=1587.6971-.016613*DA+.00000047096*DA^2
2210        B=134.8292-.00006686*DA-.00000007266*DA^2
2220        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2230     WT=GROSSWT-(CRAFT+OIL+CREW*200)
2240     RETURN
2250 !
2260 ! OH-58A (TM 55-1520-228-10   DATE:  7 APRIL 1978)
2270     CRAFT=1725 @ OIL=13 @ CREW=1 @ HELTYP$="OH-58A (T63-A-700)" @ MAXWT=3000 @ MAXTORQUE=78 ! PSI
2280        A=100574.314-335.296*TEMP
2290        B=-21976.59+33.793*TEMP
```

```
2300        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2310     YSCALE=.2*TORQUE-7
2320        A=1656.74286-.00044852*DA-.000000214579*DA^2
2330        B=179.227-.003*DA
2340        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2350     WT=GROSSWT-(CRAFT+OIL+CREW*200)
2360     RETURN
2370 !
2380 ! OH-58C (TM 55-1520-235-10   DATE:  7 APRIL 1978)
2390     CRAFT=1898 @ OIL=11 @ CREW=1 @ HELTYP$="OH-58C (T63-A-720)" @ MAXWT=3200 @ MAXTORQUE=85 ! PERCENT Q
2400        A=111904.6343-333.287*TEMP-.9386*TEMP^2
2410        B=-22630.94+50.807*TEMP
2420        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2430     YSCALE=.1*TORQUE-5
2440        A=2012.5-.00505*DA-.00000105322*DA^2+2.2958E-10*DA^3-8.3415E-15*DA^4
2450        B=367.6616-.0058745*DA-.00000047198*DA^2
2460        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2470     WT=GROSSWT-(CRAFT+OIL+CREW*200)
2480     RETURN
2490 !
2500 ! UH-1C/M (TM 55-1520-220-10   DATE:  8 SEPTEMBER 1980)
2510     CRAFT=5156 @ OIL=28 @ CREW=1 @ HELTYP$="UH-1C/M (T53-L-13)" @ MAXWT=9500 @ MAXTORQUE=50 ! PSI
2520        A=104798.2095-230.8065*TEMP-1.3629*TEMP^2
2530        B=-24682.61-2.0857*TEMP-.2726*TEMP^2+.0108*TEMP^3
2540        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2550     YSCALE=.4*TORQUE-12
2560        A=6084.1843+.02774*DA-.000002473*DA^2
2570        B=437.239-.018*DA
2580        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2590     WT=GROSSWT-(CRAFT+OIL+CREW*200)
2600     RETURN
2610 !
2620 ! UH-1D/H, EH-1H (TM 55-1520-210-10   DATE:  18 MAY 1979)
2630     CRAFT=5320 @ OIL=27 @ CREW=1 @ MAXWT=9500 @ MAXTORQUE=50 ! PSI
2640 IF H=8 THEN HELTYP$="EH-1H (T53-L-13)" ELSE HELTYP$="UH-1H (T53-L-13)"
2650        A=101054.187157-18.262435*TEMP-6.0301*TEMP^2+.14074*TEMP^3+.0020619*TEMP^4
2660        B=-24007.94261-48.213595*TEMP+1.35253*TEMP^2-.0387325*TEMP^3+.00044448*TEMP^4
2670        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2680     YSCALE=.2*TORQUE-4
2690        A=4876.42675+.0091772*DA+.00000042057*DA^2-3.2717E-12*DA^3-3.484E-15*DA^4
2700        B=861.8876-.0145625*DA-.00000028954*DA^2
2710        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2720     WT=GROSSWT-(CRAFT+OIL+CREW*200)
2730     RETURN
2740 !
2750 ! UH-60A (TM 55-1520-237-10   DATE:  21 MAY 1979)
2760     CRAFT=10984 @ OIL=15 @ CREW=2 @ HELTYP$="UH-60A (T700), TWO-ENGINE OPERATION" @ MAXWT=20250 @ MAXTORQUE=100 ! PERCENT
2770        IF TEMP(= -9 AND PA(15000 OR TEMP(= -14 AND PA(15000 THEN 2800
2780        A=114366.99834-166.684866*TEMP-28.12099*TEMP^2+1.113862*TEMP^3-.011*TEMP^4
2790        B=-23803.01534+11.427755*TEMP+6.567375*TEMP^2-.27295*TEMP^3+.00270916*TEMP^4 @ GOTO 2820
2800        A=133092.5625+646.745925*TEMP+8.2227*TEMP^2
2810        B=-27493-116.449945*TEMP-1.5191325*TEMP^2
2820        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2830     YSCALE=.1*TORQUE-4
2840        A=.09711-.001425*TEMP-.0000097213*TEMP^2
2850        B=.936644+.0012226448*TEMP-.0000310881*TEMP^2
2860     Y=A+B*YSCALE
2870        A=10745.56338+.0674077*DA-.000000153675*DA^2-2.2571E-10*DA^3
2880        B=1908.70357-.06425*DA+.000001*DA^2
2890        GROSSWT=A+B*Y @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
```

```
2900     WT=GROSSWT-(CRAFT+OIL+CREW*200)
2910     RETURN
2920 ! ********************************************************************************************************************************
****
2930 !
2940 ! PASQUILL STABILITY (from KWIK SMOKE program)
2950 !
2960 DIM Q$[6],CHART(7,9)
2961 GCLEAR @ GRAPH @ MSCALE 0,0
2962 CSIZE 9
2964 MOVE 5,55 @ LABEL "PASQUILL STABILITY PROGRAM"
2965 CSIZE 5 @ MOVE 33,9 @ LABEL "PRESS 'CONT' WHEN READY..." @ PAUSE
2966 GCLEAR
2967 PRINT " &11V" @ FOR I=1 TO 12 @ PRINT "STABILITY-";@ NEXT I @ PRINT "STABILITY";@ PRINT " &15V"
2968 PRINT " &k1S" @ PRINT "PASQUILL STABILITY INDEX" @ PRINT "_____" @ PRINT @ PRINT
2970 CLEAR @ DISP "LATITUDE OF SITE (degrees)";@ INPUT LAT
2980 DISP "DIRECTION FROM EQUATOR (N/S)";@ INPUT H$
2990 IF H$[1,1]="S" THEN LAT=-LAT
3000 DISP "LONGITUDE OF SITE (degrees)";@ INPUT LONG
3010 DISP "DIRECTION FROM GREENWICH (E/W)";@ INPUT J$
3020 IF J$[1,1]="E" THEN LONG=-LONG
3030 DISP "JULIAN DATE OF MET OBSERVATION  (001-366)";@ INPUT JDATE
3040 DISP "ZULU TIME OF MET OBSERVATION  (01-24)";@ INPUT ZTIME
3050 DISP "CEILING HEIGHT (feet)";@ INPUT CEILHT
3060 CEILHT=CEILHT*.3048
3070 DISP "CLOUD COVER (percent)";@ INPUT CLOUDS
3080 DISP "WIND SPEED (knots)";@ INPUT WIND
3090 CLEAR
3100 PRINT " &k3S" @ PRINT " &l6D";" &l7V"
3110 PRINT USING 3120 ; H$[1,1];LAT
3120 IMAGE "LATITUDE           DEG     = ",A,DDDD.D
3130 PRINT USING 3140 ; J$[1,1];LONG
3140 IMAGE "LONGITUDE          DEG     = ",A,DDDD.D
3150 PRINT USING 3160 ; JDATE
3160 IMAGE "JULIAN DATE        DAY     = ",DDDD
3170 PRINT USING 3180 ; ZTIME
3180 IMAGE "ZULU TIME          HR      = ",DDDD
3190 PRINT USING 3200 ; CEILHT
3200 IMAGE "CEILING HEIGHT     M       = ",DDDD.D
3210 PRINT USING 3220 ; CLOUDS
3220 IMAGE "CLOUD COVER        %       = ",DDDD
3230 PRINT USING 3240 ; WIND
3240 IMAGE "WIND SPEED         KTS     = ",DDDD.D
3250 ! PASQUILL STABILITY CATEGORY DATA
3260 RESTORE
3270 DATA 1,1,2,3,4,6,6
3280 DATA 1,2,2,3,4,6,6
3290 DATA 1,2,3,4,4,5,6
3300 DATA 2,2,3,4,4,5,6
3310 DATA 2,2,3,4,4,4,5
3320 DATA 2,3,3,4,4,4,5
3330 DATA 3,3,4,4,4,4,5
3340 DATA 3,3,4,4,4,4,4
3350 DATA 3,4,4,4,4,4,4
3360 FOR J=1 TO 9
3370    FOR I=1 TO 7
3380    READ CHART(I,J)
3390    NEXT I
3400 NEXT J
3410 Q$="ABCDEF"
```

```
1930 ! AH-1G (TM 55-1520-221-10   DATE: 18 MARCH 1980)
1940   CRAFT=6067 @ OIL=24 @ CREW=1 @ HELTYP$="AH-1G (T53-L-13B)" @ MAXWT=9500 @ MAXTORQUE=50 ! PSI
1950      A=98872.516782-177.53792*TEMP+3.058016*TEMP^2+.1009999*TEMP^3-.00465033*TEMP^4
1960      B=-23549.2176-5.53587*TEMP-1.18097*TEMP^2-.0315638*TEMP^3+.00135519*TEMP^4
1970      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
1980   YSCALE=.2*TORQUE-5
1990      A=5539.57727-.00467087*DA-.000000847021*DA^2
2000      B=812.884-.014*DA
2010      GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2020   WT=GROSSWT-(CRAFT+OIL+CREW*200)
2030   GOTO 1770
2040 !
2050 ! AH-1S (TM 55-1520-234-10   DATE: 17 NOVEMBER 1976)
2060   CRAFT=6910 @ OIL=29 @ CREW=1 @ HELTYP$="AH-1S (T53-L-703)" @ MAXWT=10000 @ MAXTORQUE=56 ! PSIG
2070      A=85128.094-391.18*TEMP+6.933*TEMP^2-.0974*TEMP^3
2080      B=-18754.672+78.948*TEMP-2.1278*TEMP^2-.02847*TEMP^3+.00094623*TEMP^4
2090      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2100   YSCALE=.2*TORQUE-5
2110      A=5587.655+.0004856*DA+.00000118276*DA^2-9.194E-11*DA^3
2120      B=797.3967-.01379*DA-.0000006807*DA^2+2.86E-11*DA^3
2130      GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2140   WT=GROSSWT-(CRAFT+OIL+CREW*200)
2150   GOTO 1770
2160 !
2170 ! CH-47A (TM 55-1520-209-10   DATE:  9 JANUARY 1979)
2180   CRAFT=17105 @ OIL=28 @ CREW=2 @ HELTYP$="CH-47A (T55-L-7/7B), TWO-ENGINE OPERATION" @ MAXWT=33000 @ MAXTORQUE=860 ! LB-FT
2190      A=180909.633-100.4284*TEMP-9.0486*TEMP^2+.2776*TEMP^3-.0026125*TEMP^4
2200      B=-26178.3089-9.0711*TEMP+1.25465*TEMP^2-.04209*TEMP^3+.00038448*TEMP^4
2210      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2220   YSCALE=.02*TORQUE-6
2230      A=17865.134-.03225*DA+.0000066797*DA^2-8.496E-10*DA^3
2240      B=2029-.0399*DA-.000002127*DA^2+2.382E-10*DA^3
2250      GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2260   WT=GROSSWT-(CRAFT+OIL+CREW*200)
2270   GOTO 1770
2280 !
2290 ! CH-47B (TM 55-1520-227-10-1   DATE:  23 AUGUST 1978)
2300   CRAFT=19410 @ OIL=28 @ CREW=2 @ HELTYP$="CH47B (T55-L-7B), TWO-ENGINE OPERATION" @ MAXWT=40000 @ MAXTORQUE=860 ! LB-FT
2310      A=180909.633-100.4284*TEMP-9.0486*TEMP^2+.2776*TEMP^3-.0026125*TEMP^4
2320      B=-26178.3089-9.0711*TEMP+1.25465*TEMP^2-.04209*TEMP^3+.00038448*TEMP^4
2330      TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2340   YSCALE=.01*TORQUE-3
2345      IF TEMP(0 THEN TEMP=ABS (TEMP)
2350      IF TEMP)10 THEN A=0 ELSE A=-.016543+.00069857*TEMP+.0000964286*TEMP^2
2360      IF TEMP)10 THEN B=1 ELSE B=.98997+.000602381*TEMP+.0000142857*TEMP^2+.0000028333*TEMP^3
2370      Y=A+B*YSCALE
2380   A=18600.2389-.00461*DA-.0000048954*DA^2
2390   B=3938.1889-.07542786*DA+.00000118474*DA^2
2400   GROSSWT=A+B*Y @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2410   WT=GROSSWT-(CRAFT+OIL+CREW*200)
2420   GOTO 1770
2430 !
2440 ! CH-47C (TM 55-1520-227-10-2   DATE:  23 AUGUST 1978)
2450   CRAFT=21791 @ OIL=28 @ CREW=2 @ HELTYP$="CH-47C (T55-L-7C), TWO-ENGINE OPERATION" @ MAXWT=40000 @ MAXTORQUE=990 ! LB-FT
2460      A=1012.8261-.039624*PA+.00000065447*PA^2
2470      B=-7.22367+.0003429*PA+7.6667E-9*PA^2
2480      TORQUE=A+B*TEMP @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2490   YSCALE=.0078*TORQUE-3.12
2500      A=-.042277-.00041146*TEMP+.0000907749*TEMP^2-.0000013535*TEMP^3
2510      B=1.06055-.003278*TEMP+.000043958*TEMP^2
2520      Y=(YSCALE-A)/B
2530   A=22070.8728-.0336509*DA-.00000170867*DA^2+3.01605E-10*DA^3
2540   B=4997.193-.114*DA
```

CHANGES TO DENSITY ALTITUDE

6/84

201

```
2570    GOTO 1770
2580 !
2590 ! CH-54A (TM 55-1520-217-10-1   DATE:  8 APRIL 1977)
2600    CRAFT=21895 @ OIL=15 @ CREW=2 @ HELTYP$="CH-54A (T73-P-1), TWO-ENGINE OPERATION" @ MAXWT=42000 @ MAXTORQUE=81.5 ! PERCENT
2610       IF TEMP(= -11 THEN 2640
2620       A=34787.769488-70.79067*TEMP+.7604*TEMP^2
2630       B=-296.62895-.2883*TEMP-.03579*TEMP^2 @ GOTO 2660
2640       A=35613.1689-1.22378*TEMP+.56865*TEMP^2
2650       B=-296.125-.166*TEMP-.007698*TEMP^2
2660       TORQUE=(PA-A)/B @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2670    YSCALE=.05*TORQUE-1
2680       A=.98398+.0004568*TEMP-.000002985 6*TEMP^2
2690       B=.935+.0012618*TEMP
2700       Y=A*YSCALE^B
2710    A=25925.7756-.167786*DA-.000006135 1*DA^2
2720    B=.4199-.00000075284*DA-5.4282E-11*DA^2
2730    GROSSWT=A*Y^B @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2740    WT=GROSSWT-(CRAFT+OIL+CREW*200)
2750    GOTO 1770
2760 !
2770 ! CH-54B (TM 55-1520-217-10-2   DATE:  15 APRIL 1977)
2780    CRAFT=22625 @ OIL=15 @ CREW=2 @ HELTYP$="CH-54B (T73-P-700), TWO-ENGINE OPERATION" @ MAXWT=47000 @ MAXTORQUE=100 ! PERCENT
2790       IF TEMP)= -14 THEN 2820
2800       A=94594.5894-3884.3305*TEMP-184.65234*TEMP^2-3.58638*TEMP^3-.02472*TEMP^4
2810       B=-21752.01913+473.1486*TEMP+16.41736*TEMP^2+.17233*TEMP^3 @ GOTO 2840
2820       A=121998.47092-95.3369*TEMP-7.31065*TEMP^2
2830       B=-26009.6487-30.9293*TEMP+.84051*TEMP^2+.102388*TEMP^3-.00172256*TEMP^4
2840       TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
2850    YSCALE=.05*TORQUE-1
2860       A=.9892857+.0004166667*TEMP-.00001382857*TEMP^2+.00000019733*TEMP^3
2870       B=.9479+.00171*TEMP-.000013371*TEMP^2
2880       Y=A*YSCALE^B
2890    A=16542.1559+.1092*DA-.00000203965*DA^2-.00000000067*DA^3
2900    B=14187.787-.3305*DA-.0000099711*DA^2+1.2315E-9*DA^3
2910    C=-1253.3072+.03978*DA+.000010865*DA^2-2.325E-10*DA^3
2920    GROSSWT=A+B*Y+C*Y^2 @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
2930    WT=GROSSWT-(CRAFT+OIL+CREW*200)
2940    GOTO 1770
2950 !
2960 ! OH-6A (TM 55-1520-214-10   DATE:  17 DECEMBER 1976)
2970    CRAFT=1050 @ OIL=6 @ CREW=2 @ HELTYP$="OH-6A (T63-A-5A/-700)" @ MAXWT=2550 @ MAXTORQUE=80.3 ! PSIG
2980       A=102996.723-304.236*TEMP
2990       B=-21965.0879998+34.059849213*TEMP-.19067595237*TEMP^2-1.889777778529E-3*TEMP^3
3000       TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
3010    YSCALE=.2*TORQUE-8
3020       A=1587.6971-.016613*DA+.00000047096*DA^2
3030       B=134.8292-.00006686*DA-.00000007266*DA^2
3040       GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
3050    WT=GROSSWT-(CRAFT+OIL+CREW*200)
3060    GOTO 1770
3070 !
3080 ! OH-58A (TM 55-1520-228-10   DATE:  7 APRIL 1978)
3090    CRAFT=1725 @ OIL=13 @ CREW=1 @ HELTYP$="OH-58A (T63-A-700)" @ MAXWT=3000 @ MAXTORQUE=78 ! PSI
3100       A=100574.314-335.296*TEMP
3110       B=-21976.59+33.793*TEMP
3120       TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
3130    YSCALE=.2*TORQUE-7
3140       A=1656.74286-.00044852*DA-.000000214579*DA^2
3150       B=179.227-.003*DA
3160       GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
3170    WT=GROSSWT-(CRAFT+OIL+CREW*200)
3180    GOTO 1770
3190 !
3200 ! OH-58C (TM 55-1520-235-10   DATE:  7 APRIL 1978)
```

```
3230        B=-22630.94+50.807*TEMP
3240        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
3250     YSCALE=.1*TORQUE-5
3260        A=2012.5-.00505*DA-.00000105322*DA^2+2.2958E-10*DA^3-8.3415E-15*DA^4
3270        B=367.6616-.0058745*DA-.00000047198*DA^2
3280        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
3290     WT=GROSSWT-(CRAFT+OIL+CREW*200)
3300     GOTO 1770
3310  !
3320  ! UH-1C/M (TM 55-1520-220-10   DATE:  8 SEPTEMBER 1980)
3330     CRAFT=5156 @ OIL=28 @ CREW=1 @ HELTYP$="UH-1C/M (T53-L-13)" @ MAXWT=9500 @ MAXTORQUE=50 ! PSI
3340        A=104798.2095-230.8065*TEMP-1.3629*TEMP^2
3350        B=-24682.61-2.0857*TEMP-.2726*TEMP^2+.0108*TEMP^3
3360        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
3370     YSCALE=.4*TORQUE-12
3380        A=6084.1843+.02774*DA-.000002473*DA^2
3390        B=437.239-.018*DA
3400        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
3410     WT=GROSSWT-(CRAFT+OIL+CREW*200)
3420     GOTO 1770
3430  !
3440  ! UH-1D/H, EH-1H (TM 55-1520-210-10   DATE:  18 MAY 1979)
3450     CRAFT=5320 @ OIL=27 @ CREW=1 @ MAXWT=9500 @ MAXTORQUE=50 ! PSI
3460  IF H=8 THEN HELTYP$="EH-1H (T53-L-13)" ELSE HELTYP$="UH-1H (T53-L-13)"
3470        A=101054.187157-18.262435*TEMP-6.0301*TEMP^2+.14874*TEMP^3-.0020619*TEMP^4
3480        B=-24007.94261-48.213595*TEMP+1.35253*TEMP^2-.038732S*TEMP^3+.00044448*TEMP^4
3490        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
3500     YSCALE=.2*TORQUE-4
3510        A=4876.42675+.0091772*DA+.00000042057*DA^2-3.2717E-12*DA^3-3.484E-15*DA^4
3520        B=861.8876-.0145625*DA-.000000289S4*DA^2
3530        GROSSWT=A+B*YSCALE @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
3540     WT=GROSSWT-(CRAFT+OIL+CREW*200)
3550     GOTO 1770
3560  !
3570  ! UH-60A (TM 55-1520-237-10   DATE:  21 MAY 1979)
3580     CRAFT=10984 @ OIL=15 @ CREW=2 @ HELTYP$="UH-60A (T700), TWO-ENGINE OPERATION" @ MAXWT=20250 @ MAXTORQUE=100 ! PERCENT
3590        IF TEMP(= -9 AND PA(= 15000 OR TEMP(= -14 AND PA)= 15000 THEN 3620
3600        A=114366.99834-166.684866*TEMP-28.12099*TEMP^2+1.113862*TEMP^3-.011*TEMP^4
3610        B=-23803.01534+11.427755*TEMP+6.567375*TEMP^2-.27295*TEMP^3+.00270916*TEMP^4 @ GOTO 3640
3620        A=133092.5625+646.745925*TEMP+8.2227*TEMP^2
3630        B=-27493-116.449945*TEMP-1.5191325*TEMP^2
3640        TORQUE=EXP ((PA-A)/B) @ IF TORQUE)MAXTORQUE THEN TORQUE=MAXTORQUE
3650     YSCALE=.1*TORQUE-4
3660        A=.09711-.001425*TEMP-.0000097213*TEMP^2
3670        B=.936644+.0012226448*TEMP-.00000310881*TEMP^2
3680     Y=A+B*YSCALE
3690        A=10745.56338+.0674077*DA-.000000153675*DA^2-2.2571E-10*DA^3
3700        B=1908.70357-.06425*DA+.000001*DA^2
3710        GROSSWT=A+B*Y @ IF GROSSWT)MAXWT THEN GROSSWT=MAXWT
3720     WT=GROSSWT-(CRAFT+OIL+CREW*200)
3730     GOTO 1770
3740  ! ********************************************************************************************************************************************
****
3750  END
```